AxonASP Documentation

Reference Manual & User Guide

G3pix

2026

Table of Contents

1 Welcome to AxonASP 2.0: The Ultimate Classic ASP Engine for the Modern Web 153

1.1 What Is AxonASP? 153

1.2 Why Choose AxonASP? The Performance Revolution 153

1.3 What's New in Version 2.0? 154

1.4 Understanding This Manual 155

1.5 Key Capabilities at a Glance 156

1.6 Getting Started 157

1.7 A Word About Version 2.0 157

1.8 Questions? Ready to Deploy? 157

2 Understand the AxonASP Project Structure 157

2.1 Overview 157

2.2 Building 157

2.3 Project Directory Structure 158

2.4 Required Runtime Files 159

2.5 Environment Variable Override 160

2.6 Service Ports and Endpoints 160

2.7 Docker Support 160

2.8 Deployment Architecture 161

2.9 Deployment Architecture 161

2.10 Configuration Reference 162

2.11 Remarks 162

3 Understand AxonASP Internal Architecture (Go VM and Compiler) 162

3.1 Overview 162

3.2 Origin Story and Mission 162

3.3 Core Engine Design 163

3.4 High-Level Pipeline 163

3.5 Runtime Modes Share One Core 163

3.6 Go Implementation Strategy and Optimizations 163

3.7 Wrapping Go Libraries as ASP Objects 163
3.8 Performance Discipline in Go 164
3.9 Compiler Architecture (No AST Rule) 164
3.10 Rule: No AST 164
3.11 What Replaces AST 164
3.12 Compiler Tooling for Runtime Efficiency 164
3.13 VM Architecture 165
3.14 VM Type 165
3.15 Opcode Model 165
3.16 Call Frames and ByRef 165
3.17 Intrinsic ASP Object Emulation 165
3.18 Type System and Memory Model 166
3.19 Variant Representation in Go 166
3.20 Compatibility Semantics 166
3.21 Native Object and Library Lifecycle 166
3.22 Creation 166
3.23 Dispatch 166
3.24 Cleanup and Lifetime 166
3.25 Practical Extension Guide 167
3.26 Add a New Native Library (Recommended Pattern) 167
3.27 Add Compiler/VM Features Safely 167
3.28 Internal Files to Know First 167
3.29 Design Principles Summary 167
4 Use axonasp-testsuite 168
4.1 Overview 168
4.2 Discovery Rules 168
4.3 Command-Line Usage 168
4.4 How Test Files Are Executed 168
4.5 Creating ASP Tests 169
4.6 Assertion API 169
4.7 Suite Properties 171
4.8 Practical Test Organization 171

4.9 Example: Multi-Assertion Test File 171
4.10 CI Usage 172
4.11 Runtime Integration Notes 172
4.12 Remarks 172
5 global.asa 173
5.1 Overview 173
5.2 File Location 173
5.3 Supported Event Handlers 173
5.4 Static Object Declarations 174
5.5 What Is Not Supported 174
5.6 Remarks 175
5.7 Code Example 175
6 web.config Support 176
6.1 Overview 176
6.2 Prerequisites 176
6.3 Supported Sections 176
6.4 Remarks 180
6.5 Code Example 180
7 MyInfo.xml 181
7.1 Overview 181
7.2 File Location 181
7.3 File Format 181
7.4 Usage 181
7.5 Properties 182
7.6 Remarks 182
7.7 Code Example 182
8 Script Caching 183
8.1 Overview 183
8.2 Cache Tiers 183
8.3 Configuration Reference 185
8.4 Remarks 185
9 Locale Support and Language Formatting 186

9.1 Overview 186
9.2 Working with Locales 186
9.3 Supported Locales and Variants 187
9.4 Formatting Examples 189
9.5 Affected Functions 191
9.6 Performance Notes 191
9.7 Configuration 192
9.8 Troubleshooting 192
10 FastCGI Setup 192
10.1 Overview 192
10.2 Prerequisites 193
10.3 DOCUMENT_ROOT and SCRIPT_NAME Parameters 193
10.4 Nginx with FastCGI 193
10.5 Key FastCGI Parameters 195
10.6 Apache with FastCGI 196
10.7 IIS with FastCGI 197
10.8 Unix Socket Mode 197
10.9 Remarks 198
11 Reverse Proxy Configuration 198
11.1 Overview 198
11.2 Prerequisites 198
11.3 Why Not Expose AxonASP Directly 198
11.4 Nginx (Proxy Mode) 199
11.5 Apache (Proxy Mode) 199
11.6 Caddy (Proxy Mode) 200
11.7 IIS (Proxy Mode) 200
11.8 Load Balancing Multiple Instances 201
11.9 Remarks 201
12 Running AxonASP as a Linux Service 202
12.1 Overview 202
12.2 Prerequisites 202
12.3 Building for Linux 202

12.4 Creating the Service User 202
12.5 Deploying Files 202
12.6 Creating the systemd Unit File 203
12.7 Enabling and Starting the Service 203
12.8 Managing the Service 204
12.9 Configuring the Web Root and Port 204
12.10 Nginx Reverse Proxy Integration 204
12.11 Remarks 205
13 Installing AxonASP from Linux Packages 205
13.1 Overview 205
13.2 Prerequisites 205
13.3 Downloading a Package 205
13.4 Installing the Package 206
13.5 What Gets Installed 206
13.6 Symbolic Links 207
13.7 Starting the Server After Installation 208
13.8 Remarks 208
14 AxonASP Service Wrapper 208
14.1 Overview 208
14.2 Binary Name 209
14.3 How It Works 209
14.4 Install and Manage the Service 209
14.5 Windows 209
14.6 Linux or macOS 209
14.7 Automated Installation and Uninstallation Scripts 210
14.8 Service Configuration Flags in axonasp.toml 210
14.9 Deployment Recommendations 211
15 MCP Server and VS Code Integration 212
15.1 Overview 212
15.2 Starting the MCP Server 212
15.3 Configuring VS Code 212
15.4 SSE Mode 213

15.5 Remarks 213
16 System Pages and Error Pages 213
16.1 Overview 213
16.2 axonasp-pages/ 214
16.3 error-pages/ 215
16.4 Remarks 216
17 AxonASP Configuration Reference 217
17.1 Overview 217
17.2 Configuration Location 217
17.3 Environment Variable Override 217
17.4 Global Settings [global] 218
17.5 CLI Settings [cli] 224
17.6 Web Server Settings [server] 225
17.7 FastCGI Server Settings [fastcgi] 228
17.8 Database Configuration [g3db] 229
17.9 Mail Configuration [g3mail] 230
17.10 AxonASP Functions Settings [axfunctions] 231
17.11 MCP Server Settings [mcp] 232
17.12 MSWC Component Settings [mswc] 233
17.13 Service Wrapper Settings [service] 234
17.14 Configuration Examples 236
17.15 Best Practices 237
17.16 Code Example 238
18 The #include Directive 238
18.1 Overview 238
18.2 Syntax 238
18.3 Parameters and Arguments 238
18.4 How it Works 238
18.5 Remarks 239
18.6 Code Example 239
19 ASP Variables 239
19.1 Overview 239

19.2 Syntax 240
19.3 How it Works 240
19.4 API Reference 240
19.5 Remarks 240
19.6 Code Example 241
20 ASP Procedures 241
20.1 Overview 241
20.2 Syntax 241
20.3 How it Works 242
20.4 API Reference 242
20.5 Remarks 242
20.6 Code Example 243
21 ASP Conditionals 243
21.1 Overview 243
21.2 Syntax 243
21.3 How it Works 244
21.4 API Reference 244
21.5 Remarks 244
21.6 Code Example 245
22 ASP Looping 245
22.1 Overview 245
22.2 Syntax 245
22.3 How it Works 246
22.4 API Reference 246
22.5 Remarks 246
22.6 Code Example 247
23 ASP Intrinsic Objects - Quick Reference 247
23.1 Overview 247
23.2 At a Glance 248
23.3 Request 248
23.4 Response 249
23.5 Session 249

23.6 Application 250
23.7 Server 250
23.8 ASPError 250
23.9 Detailed Pages 251
24 ASP Quick Reference 251
24.1 Intrinsic Objects 251
24.2 Runtime Pattern 251
24.3 Minimal ASP Page 251
25 Write ASP Script Syntax 252
25.1 Overview 252
25.2 Prerequisites 252
25.3 Core Delimiters 252
25.4 Write Output with Response.Write 252
25.5 Use Page Directives 253
25.6 Add Comments 253
25.7 Basic Control Flow 253
25.8 How It Works 254
25.9 API Reference 254
26 Handle ASP Form Data 254
26.1 Overview 254
26.2 Prerequisites 254
26.3 Read a Single Field 254
26.4 Read Multi-Value Fields 255
26.5 Enumerate Posted Keys 255
26.6 How It Works 256
26.7 API Reference 256
27 Use ASP Cookies 256
27.1 Overview 256
27.2 Prerequisites 256
27.3 How it Works 257
27.4 API Reference 257
27.5 Code Example 257

27.6 Remarks 258
28 Request Object 258
28.1 Overview 258
28.2 Collections 258
28.3 Properties 261
28.4 Methods 262
28.5 Remarks 262
29 Response Object 263
29.1 Overview 263
29.2 Methods 263
29.3 Properties 267
29.4 Remarks 271
30 Server Object 271
30.1 Overview 271
30.2 Methods 272
30.3 Properties 276
30.4 Remarks 277
31 Manage Session State 277
31.1 Overview 277
31.2 Prerequisites 277
31.3 Store and Read Values 277
31.4 Methods 277
31.5 Properties 278
31.6 How It Works 280
31.7 Best Practices 280
32 Use the ASP Application Object 280
32.1 Overview 280
32.2 Prerequisites 281
32.3 How it Works 281
32.4 API Reference 281
32.5 Code Example 281
32.6 Remarks 282

33 Use the ASPError Object 282
33.1 Overview 282
33.2 Prerequisites 282
33.3 How it Works 282
33.4 API Reference 282
33.5 Code Example 283
33.6 Remarks 283
34 Use the ASP Dictionary Object 283
34.1 Overview 283
34.2 Syntax 284
34.3 How it Works 284
34.4 API Reference 284
34.5 Code Example 284
34.6 Remarks 285
35 Use G3Pix AxonASP Library Objects 285
35.1 Overview 285
35.2 Prerequisites 285
35.3 Syntax 286
35.4 Object Creation Summary 286
35.5 Return Value 287
35.6 Remarks 287
35.7 Code Example 287
35.8 API Reference 287
36 Use the G3AXON.FUNCTIONS Library 288
36.1 Overview 288
36.2 Syntax 288
36.3 Prerequisites 288
36.4 How it Works 288
36.5 API Reference 288
36.6 Code Example 288
37 G3AXON.FUNCTIONS Method Reference 289
37.1 Overview 289

37.2 System and Environment 289
37.3 Configuration 291
37.4 Math and Numeric 291
37.5 Array 292
37.6 String 293
37.7 Hash and Encoding 294
37.8 Validation and Type Inspection 294
37.9 Date, Time, and Network 295
37.10 Remarks 295
38 Properties 296
38.1 Overview 296
38.2 Property List 296
38.3 Remarks 296
39 AxEngineName 296
39.1 Overview 296
39.2 Prerequisites 296
39.3 Syntax 296
39.4 Parameters 296
39.5 Return Value 296
39.6 Remarks 296
39.7 Example 296
39.8 API Reference 297
40 AxVersion 297
40.1 Overview 297
40.2 Prerequisites 297
40.3 Syntax 297
40.4 Parameters 297
40.5 Return Value 297
40.6 Remarks 297
40.7 Example 297
40.8 API Reference 298
41 Get Runtime Diagnostic Report 298

41.1 Overview 298
41.2 Prerequisites 298
41.3 Syntax 298
41.4 Parameters 298
41.5 Return Value 298
41.6 Remarks 298
41.7 Example 299
41.8 API Reference 299
41.9 Parameters 299
41.10 Return Value 299
41.11 Remarks 300
41.12 Example 300
41.13 API Reference 300
42 AxGetConfig 300
42.1 Overview 300
42.2 Prerequisites 300
42.3 Syntax 300
42.4 Parameters 300
42.5 Return Value 301
42.6 Remarks 301
42.7 Example 301
42.8 API Reference 301
43 AxGetConfigKeys 302
43.1 Overview 302
43.2 Prerequisites 302
43.3 Syntax 302
43.4 Parameters 302
43.5 Return Value 302
43.6 Remarks 302
43.7 Example 302
43.8 API Reference 303
44 Get the User Home Directory Path 303

44.1 Overview 303
44.2 Prerequisites 303
44.3 Syntax 303
44.4 Parameters 303
44.5 Return Value 303
44.6 Remarks 303
44.7 Example 303
44.8 API Reference 304
45 Get the Configuration File Path 304
45.1 Overview 304
45.2 Prerequisites 304
45.3 Syntax 304
45.4 Parameters 304
45.5 Return Value 304
45.6 Remarks 304
45.7 Example 304
45.8 API Reference 305
46 Get the Cache Directory Path 305
46.1 Overview 305
46.2 Prerequisites 305
46.3 Syntax 305
46.4 Parameters 305
46.5 Return Value 305
46.6 Remarks 305
46.7 Example 305
46.8 API Reference 306
47 Check If a Character Is a Path Separator 306
47.1 Overview 306
47.2 Prerequisites 306
47.3 Syntax 306
47.4 Parameters 306
47.5 Return Value 306

47.6 Remarks 306
47.7 Example 306
47.8 API Reference 307
48 Change File Timestamps 307
48.1 Overview 307
48.2 Prerequisites 307
48.3 Syntax 307
48.4 Parameters 307
48.5 Return Value 308
48.6 Remarks 308
48.7 Example 308
48.8 API Reference 308
49 Change File Mode Permissions 308
49.1 Overview 308
49.2 Prerequisites 308
49.3 Syntax 308
49.4 Parameters 309
49.5 Return Value 309
49.6 Remarks 309
49.7 Example 309
49.8 API Reference 309
50 Create a Hard Link 310
50.1 Overview 310
50.2 Prerequisites 310
50.3 Syntax 310
50.4 Parameters 310
50.5 Return Value 310
50.6 Remarks 310
50.7 Example 310
50.8 API Reference 311
51 Change File Owner and Group 311
51.1 Overview 311

51.2 Prerequisites 311
51.3 Syntax 311
51.4 Parameters 311
51.5 Return Value 311
51.6 Remarks 311
51.7 Example 312
51.8 API Reference 312
52 Shut Down the AxonASP Server 312
52.1 Overview 312
52.2 Prerequisites 312
52.3 Syntax 312
52.4 Parameters 312
52.5 Return Value 313
52.6 Remarks 313
52.7 Example 313
52.8 API Reference 313
53 Change the Current Working Directory 314
53.1 Overview 314
53.2 Prerequisites 314
53.3 Syntax 314
53.4 Parameters 314
53.5 Return Value 314
53.6 Remarks 314
53.7 Example 314
53.8 API Reference 315
53.9 Parameters 315
53.10 Return Value 315
53.11 Remarks 315
53.12 Example 315
53.13 API Reference 316
54 Get the Machine Hostname 316
54.1 Overview 316

54.2 Prerequisites 316
54.3 Syntax 316
54.4 Parameters 316
54.5 Return Value 316
54.6 Remarks 316
54.7 Example 316
54.8 API Reference 317
55 Clear All Environment Variables 317
55.1 Overview 317
55.2 Prerequisites 317
55.3 Syntax 317
55.4 Parameters 317
55.5 Return Value 317
55.6 Remarks 317
55.7 Example 317
55.8 API Reference 318
55.9 Parameters 318
55.10 Return Value 318
55.11 Remarks 318
55.12 Example 318
55.13 API Reference 319
56 Get an Environment Variable Value 319
56.1 Overview 319
56.2 Prerequisites 319
56.3 Syntax 319
56.4 Parameters 319
56.5 Return Value 319
56.6 Remarks 320
56.7 Example 320
56.8 API Reference 320
57 Get the Current Process ID 320
57.1 Overview 320

57.2 Prerequisites 320
57.3 Syntax 320
57.4 Parameters 320
57.5 Return Value 320
57.6 Remarks 321
57.7 Example 321
57.8 API Reference 321
58 Get the Effective User ID 321
58.1 Overview 321
58.2 Prerequisites 321
58.3 Syntax 321
58.4 Parameters 321
58.5 Return Value 321
58.6 Remarks 322
58.7 Example 322
58.8 API Reference 322
59 Get the Directory Separator Character 322
59.1 Overview 322
59.2 Prerequisites 322
59.3 Syntax 322
59.4 Parameters 322
59.5 Return Value 323
59.6 Remarks 323
59.7 Example 323
59.8 API Reference 323
60 Get the Path List Separator Character 323
60.1 Overview 323
60.2 Prerequisites 323
60.3 Syntax 323
60.4 Parameters 323
60.5 Return Value 324
60.6 Remarks 324

60.7 Example 324
60.8 API Reference 324
61 Get the Native Integer Size in Bytes 324
61.1 Overview 324
61.2 Prerequisites 324
61.3 Syntax 324
61.4 Parameters 324
61.5 Return Value 325
61.6 Remarks 325
61.7 Example 325
61.8 API Reference 325
62 Get the Platform Architecture Bit Width 325
62.1 Overview 325
62.2 Prerequisites 325
62.3 Syntax 325
62.4 Parameters 325
62.5 Return Value 325
62.6 Remarks 326
62.7 Example 326
62.8 API Reference 326
63 Get the Executable Path 326
63.1 Overview 326
63.2 Prerequisites 326
63.3 Syntax 326
63.4 Parameters 326
63.5 Return Value 326
63.6 Remarks 327
63.7 Example 327
63.8 API Reference 327
64 Execute a Shell Command 327
64.1 Overview 327
64.2 Prerequisites 327

64.3 Syntax 327
64.4 Parameters 327
64.5 Return Value 327
64.6 Remarks 328
64.7 Example 328
64.8 API Reference 328
65 Get System Information 328
65.1 Overview 328
65.2 Prerequisites 328
65.3 Syntax 328
65.4 Parameters 328
65.5 Return Value 329
65.6 Remarks 329
65.7 Example 329
65.8 API Reference 330
66 Get the Current Process User Name 330
66.1 Overview 330
66.2 Prerequisites 330
66.3 Syntax 330
66.4 Parameters 330
66.5 Return Value 330
66.6 Remarks 330
66.7 Example 330
66.8 API Reference 331
67 Write HTML-Escaped Text to the Response 331
67.1 Overview 331
67.2 Prerequisites 331
67.3 Syntax 331
67.4 Parameters 331
67.5 Return Value 331
67.6 Remarks 331
67.7 Example 332

67.8 API Reference 332
68 Return the Maximum Value 332
68.1 Overview 332
68.2 Prerequisites 332
68.3 Syntax 332
68.4 Parameters 332
68.5 Return Value 333
68.6 Remarks 333
68.7 Example 333
68.8 API Reference 333
69 Return the Minimum Value 333
69.1 Overview 333
69.2 Prerequisites 333
69.3 Syntax 333
69.4 Parameters 333
69.5 Return Value 334
69.6 Remarks 334
69.7 Example 334
69.8 API Reference 334
70 Get the Maximum Platform Integer Value 334
70.1 Overview 334
70.2 Prerequisites 335
70.3 Syntax 335
70.4 Parameters 335
70.5 Return Value 335
70.6 Remarks 335
70.7 Example 335
70.8 API Reference 335
71 Get the Minimum Platform Integer Value 335
71.1 Overview 335
71.2 Prerequisites 335
71.3 Syntax 336

71.4 Parameters 336
71.5 Return Value 336
71.6 Remarks 336
71.7 Example 336
71.8 API Reference 336
72 Round a Number Up 336
72.1 Overview 336
72.2 Prerequisites 336
72.3 Syntax 336
72.4 Parameters 337
72.5 Return Value 337
72.6 Remarks 337
72.7 Example 337
72.8 API Reference 337
73 Round a Number Down 337
73.1 Overview 337
73.2 Prerequisites 337
73.3 Syntax 338
73.4 Parameters 338
73.5 Return Value 338
73.6 Remarks 338
73.7 Example 338
73.8 API Reference 338
74 Generate a Random Integer 338
74.1 Overview 338
74.2 Prerequisites 339
74.3 Syntax 339
74.4 Parameters 339
74.5 Return Value 339
74.6 Remarks 339
74.7 Example 340
74.8 API Reference 340

75 Format a Number with Grouping and Decimal Options 340
75.1 Overview 340
75.2 Prerequisites 340
75.3 Syntax 340
75.4 Parameters 340
75.5 Return Value 341
75.6 Remarks 341
75.7 Example 341
75.8 API Reference 342
76 Return the Constant Pi 342
76.1 Overview 342
76.2 Prerequisites 342
76.3 Syntax 342
76.4 Parameters 342
76.5 Return Value 342
76.6 Remarks 342
76.7 Example 342
76.8 API Reference 343
77 Get the Smallest Positive Float Value 343
77.1 Overview 343
77.2 Prerequisites 343
77.3 Syntax 343
77.4 Parameters 343
77.5 Return Value 343
77.6 Remarks 343
77.7 Example 343
77.8 API Reference 344
78 Get the Float Precision Digit Count 344
78.1 Overview 344
78.2 Prerequisites 344
78.3 Syntax 344
78.4 Parameters 344

78.5 Return Value 344
78.6 Remarks 344
78.7 Example 344
78.8 API Reference 345
79 Count Elements in an Array 345
79.1 Overview 345
79.2 Prerequisites 345
79.3 Syntax 345
79.4 Parameters 345
79.5 Return Value 345
79.6 Remarks 345
79.7 Example 345
79.8 API Reference 346
80 Split a String into an Array 346
80.1 Overview 346
80.2 Prerequisites 346
80.3 Syntax 346
80.4 Parameters 346
80.5 Return Value 346
80.6 Remarks 347
80.7 Example 347
80.8 API Reference 347
81 Join Array Elements into a String 347
81.1 Overview 347
81.2 Prerequisites 347
81.3 Syntax 348
81.4 Parameters 348
81.5 Return Value 348
81.6 Remarks 348
81.7 Example 348
81.8 API Reference 348
82 Reverse an Array 349

82.1 Overview 349
82.2 Prerequisites 349
82.3 Syntax 349
82.4 Parameters 349
82.5 Return Value 349
82.6 Remarks 349
82.7 Example 349
82.8 API Reference 350
83 Create a Numeric Range Array 350
83.1 Overview 350
83.2 Prerequisites 350
83.3 Syntax 350
83.4 Parameters 350
83.5 Return Value 350
83.6 Remarks 351
83.7 Example 351
83.8 API Reference 351
84 Replace All Occurrences of a Substring 351
84.1 Overview 351
84.2 Prerequisites 351
84.3 Syntax 352
84.4 Parameters 352
84.5 Return Value 352
84.6 Remarks 352
84.7 Example 352
84.8 API Reference 353
85 Pad a String to a Specified Length 353
85.1 Overview 353
85.2 Prerequisites 353
85.3 Syntax 353
85.4 Parameters 353
85.5 Return Value 354

85.6 Remarks 354
85.7 Example 354
85.8 API Reference 354
86 Repeat a String 354
86.1 Overview 354
86.2 Prerequisites 355
86.3 Syntax 355
86.4 Parameters 355
86.5 Return Value 355
86.6 Remarks 355
86.7 Example 355
86.8 API Reference 355
87 Uppercase the First Character 356
87.1 Overview 356
87.2 Prerequisites 356
87.3 Syntax 356
87.4 Parameters 356
87.5 Return Value 356
87.6 Remarks 356
87.7 Example 356
87.8 API Reference 357
88 Count Words in a String 357
88.1 Overview 357
88.2 Prerequisites 357
88.3 Syntax 357
88.4 Parameters 357
88.5 Return Value 357
88.6 Remarks 357
88.7 Example 358
88.8 API Reference 358
89 Convert Newlines to HTML Line Breaks 358
89.1 Overview 358

89.2 Prerequisites 358
89.3 Syntax 358
89.4 Parameters 358
89.5 Return Value 359
89.6 Remarks 359
89.7 Example 359
89.8 API Reference 359
90 Trim Characters from Both Ends of a String 359
90.1 Overview 359
90.2 Prerequisites 359
90.3 Syntax 360
90.4 Parameters 360
90.5 Return Value 360
90.6 Remarks 360
90.7 Example 360
90.8 API Reference 360
91 Parse a CSV Row String 361
91.1 Overview 361
91.2 Prerequisites 361
91.3 Syntax 361
91.4 Parameters 361
91.5 Return Value 361
91.6 Remarks 361
91.7 Example 362
91.8 API Reference 362
92 Compute an MD5 Hash 362
92.1 Overview 362
92.2 Prerequisites 362
92.3 Syntax 362
92.4 Parameters 362
92.5 Return Value 362
92.6 Remarks 363

92.7 Example 363
92.8 API Reference 363
93 Compute a SHA-1 Hash 363
93.1 Overview 363
93.2 Prerequisites 363
93.3 Syntax 363
93.4 Parameters 364
93.5 Return Value 364
93.6 Remarks 364
93.7 Example 364
93.8 API Reference 364
94 Compute a String Hash by Algorithm 364
94.1 Overview 364
94.2 Prerequisites 364
94.3 Syntax 365
94.4 Parameters 365
94.5 Return Value 365
94.6 Remarks 365
94.7 Example 365
94.8 API Reference 366
95 Encode a String to Base64 366
95.1 Overview 366
95.2 Prerequisites 366
95.3 Syntax 366
95.4 Parameters 366
95.5 Return Value 366
95.6 Remarks 366
95.7 Example 366
95.8 API Reference 367
96 Decode a Base64 String 367
96.1 Overview 367
96.2 Prerequisites 367

96.3 Syntax 367
96.4 Parameters 367
96.5 Return Value 367
96.6 Remarks 367
96.7 Example 367
96.8 API Reference 368
97 Decode a URL-Encoded String 368
97.1 Overview 368
97.2 Prerequisites 368
97.3 Syntax 368
97.4 Parameters 368
97.5 Return Value 368
97.6 Remarks 368
97.7 Example 369
97.8 API Reference 369
98 Decode a Raw URL-Encoded String 369
98.1 Overview 369
98.2 Prerequisites 369
98.3 Syntax 369
98.4 Parameters 369
98.5 Return Value 370
98.6 Remarks 370
98.7 Example 370
98.8 API Reference 370
99 Convert RGB Values to a Hexadecimal Color Code 370
99.1 Overview 370
99.2 Prerequisites 370
99.3 Syntax 370
99.4 Parameters 371
99.5 Return Value 371
99.6 Remarks 371
99.7 Example 371

99.8 API Reference 372
100 Convert a Hexadecimal Color Code to RGB 372
100.1 Overview 372
100.2 Prerequisites 372
100.3 Syntax 372
100.4 Parameters 372
100.5 Return Value 372
100.6 Remarks 372
100.7 Example 373
100.8 API Reference 373
101 Get the Default Logo as a Base64 Data URI 373
101.1 Overview 373
101.2 Prerequisites 373
101.3 Syntax 373
101.4 Parameters 374
101.5 Return Value 374
101.6 Remarks 374
101.7 Example 374
101.8 API Reference 374
102 Escape HTML Special Characters 375
102.1 Overview 375
102.2 Prerequisites 375
102.3 Syntax 375
102.4 Parameters 375
102.5 Return Value 375
102.6 Remarks 375
102.7 Example 375
102.8 API Reference 375
103 Strip HTML and XML Tags from a String 376
103.1 Overview 376
103.2 Prerequisites 376
103.3 Syntax 376

103.4 Parameters 376
103.5 Return Value 376
103.6 Remarks 376
103.7 Example 376
103.8 API Reference 377
104 Validate an IP Address 377
104.1 Overview 377
104.2 Prerequisites 377
104.3 Syntax 377
104.4 Parameters 377
104.5 Return Value 377
104.6 Remarks 377
104.7 Example 378
104.8 API Reference 378
105 Validate an Email Address 378
105.1 Overview 378
105.2 Prerequisites 378
105.3 Syntax 378
105.4 Parameters 378
105.5 Return Value 379
105.6 Remarks 379
105.7 Example 379
105.8 API Reference 379
106 Check If a Value Is an Integer 379
106.1 Overview 379
106.2 Prerequisites 380
106.3 Syntax 380
106.4 Parameters 380
106.5 Return Value 380
106.6 Remarks 380
106.7 Example 380
106.8 API Reference 380

107 Check If a Value Is a Floating-Point Number 381
107.1 Overview 381
107.2 Prerequisites 381
107.3 Syntax 381
107.4 Parameters 381
107.5 Return Value 381
107.6 Remarks 381
107.7 Example 381
107.8 API Reference 381
108 Check If a String Contains Only Alphabetic Characters 382
108.1 Overview 382
108.2 Prerequisites 382
108.3 Syntax 382
108.4 Parameters 382
108.5 Return Value 382
108.6 Remarks 382
108.7 Example 382
108.8 API Reference 383
109 Check If a String Contains Only Alphanumeric Characters 383
109.1 Overview 383
109.2 Prerequisites 383
109.3 Syntax 383
109.4 Parameters 383
109.5 Return Value 383
109.6 Remarks 383
109.7 Example 383
109.8 API Reference 384
110 Check If a Value Is Empty 384
110.1 Overview 384
110.2 Prerequisites 384
110.3 Syntax 384
110.4 Parameters 384

110.5 Return Value 384
110.6 Remarks 384
110.7 Example 385
110.8 API Reference 385
110.9 Parameters 385
110.10 Return Value 386
110.11 Remarks 386
110.12 Example 386
110.13 API Reference 386
111 Get the Current Unix Timestamp 386
111.1 Overview 386
111.2 Prerequisites 386
111.3 Syntax 387
111.4 Parameters 387
111.5 Return Value 387
111.6 Remarks 387
111.7 Example 387
111.8 API Reference 387
112 Format a Date and Time String 387
112.1 Overview 387
112.2 Prerequisites 387
112.3 Syntax 388
112.4 Parameters 388
112.5 Return Value 389
112.6 Remarks 389
112.7 Example 389
112.8 API Reference 389
113 Get the Last Modified Timestamp of the Current Page 389
113.1 Overview 389
113.2 Prerequisites 389
113.3 Syntax 389
113.4 Parameters 390

113.5 Return Value 390
113.6 Remarks 390
113.7 Example 390
113.8 API Reference 390
114 Fetch the Content of a Remote URL 391
114.1 Overview 391
114.2 Prerequisites 391
114.3 Syntax 391
114.4 Parameters 391
114.5 Return Value 391
114.6 Remarks 391
114.7 Example 391
114.8 API Reference 392
115 Generate a GUID 392
115.1 Overview 392
115.2 Prerequisites 392
115.3 Syntax 392
115.4 Parameters 392
115.5 Return Value 392
115.6 Remarks 392
115.7 Example 393
115.8 API Reference 393
116 Get the Default CSS Stylesheet Content 393
116.1 Overview 393
116.2 Prerequisites 393
116.3 Syntax 393
116.4 Parameters 393
116.5 Return Value 393
116.6 Remarks 394
116.7 Example 394
116.8 API Reference 394
117 Use the G3CRYPTO Library 394

117.1 Overview 394
117.2 Prerequisites 394
117.3 Syntax 394
117.4 How it Works 395
117.5 API Reference 395
117.6 Code Example 395
118 G3CRYPTO Methods 396
118.1 Overview 396
118.2 Methods 396
118.3 Remarks 398
119 G3CRYPTO Properties 398
119.1 Overview 398
119.2 Properties 398
119.3 Remarks 398
120 Compute a BLAKE2b-256 Digest 398
120.1 Overview 398
120.2 Prerequisites 398
120.3 Syntax 399
120.4 Parameters 399
120.5 Return Value 399
120.6 Remarks 399
120.7 Example 399
120.8 API Reference 399
121 Compute a BLAKE2b-512 Digest 400
121.1 Overview 400
121.2 Prerequisites 400
121.3 Syntax 400
121.4 Parameters 400
121.5 Return Value 400
121.6 Remarks 400
121.7 Example 400
121.8 API Reference 400

122 Compute a Hash as a Byte Array 401
122.1 Overview 401
122.2 Prerequisites 401
122.3 Syntax 401
122.4 Parameters 401
122.5 Return Value 401
122.6 Remarks 401
122.7 Example 402
122.8 API Reference 402
123 Get the bcrypt Cost 402
123.1 Overview 402
123.2 Prerequisites 402
123.3 Syntax 402
123.4 Parameters 402
123.5 Return Value 402
123.6 Remarks 402
123.7 Example 403
123.8 API Reference 403
124 Hash a Password with bcrypt 403
124.1 Overview 403
124.2 Prerequisites 403
124.3 Syntax 403
124.4 Parameters 403
124.5 Return Value 403
124.6 Remarks 404
124.7 Example 404
124.8 API Reference 404
125 Compute an HMAC-SHA256 Signature 404
125.1 Overview 404
125.2 Prerequisites 404
125.3 Syntax 404
125.4 Parameters 404

125.5 Return Value 405
125.6 Remarks 405
125.7 Example 405
125.8 API Reference 405
126 Compute an HMAC-SHA512 Signature 405
126.1 Overview 405
126.2 Prerequisites 405
126.3 Syntax 405
126.4 Parameters 405
126.5 Return Value 406
126.6 Remarks 406
126.7 Example 406
126.8 API Reference 406
127 Reset the Internal Hash State 406
127.1 Overview 406
127.2 Prerequisites 406
127.3 Syntax 406
127.4 Parameters 407
127.5 Return Value 407
127.6 Remarks 407
127.7 Example 407
127.8 API Reference 407
128 Compute an MD5 Digest 407
128.1 Overview 407
128.2 Prerequisites 407
128.3 Syntax 407
128.4 Parameters 408
128.5 Return Value 408
128.6 Remarks 408
128.7 Example 408
128.8 API Reference 408
129 Derive a Key with PBKDF2-SHA256 408

129.1 Overview 408
129.2 Prerequisites 409
129.3 Syntax 409
129.4 Parameters 409
129.5 Return Value 409
129.6 Remarks 409
129.7 Example 409
129.8 API Reference 410
130 Generate a Secure Random Base64 String 410
130.1 Overview 410
130.2 Prerequisites 410
130.3 Syntax 410
130.4 Parameters 410
130.5 Return Value 410
130.6 Remarks 411
130.7 Example 411
130.8 API Reference 411
131 Generate Secure Random Bytes 411
131.1 Overview 411
131.2 Prerequisites 411
131.3 Syntax 411
131.4 Parameters 411
131.5 Return Value 412
131.6 Remarks 412
131.7 Example 412
131.8 API Reference 412
132 Generate a Secure Random Hex String 412
132.1 Overview 412
132.2 Prerequisites 412
132.3 Syntax 412
132.4 Parameters 412
132.5 Return Value 413

132.6 Remarks 413
132.7 Example 413
132.8 API Reference 413
133 Set the bcrypt Cost 413
133.1 Overview 413
133.2 Prerequisites 413
133.3 Syntax 414
133.4 Parameters 414
133.5 Return Value 414
133.6 Remarks 414
133.7 Example 414
133.8 API Reference 414
134 Compute a SHA-1 Digest 415
134.1 Overview 415
134.2 Prerequisites 415
134.3 Syntax 415
134.4 Parameters 415
134.5 Return Value 415
134.6 Remarks 415
134.7 Example 415
134.8 API Reference 416
135 Compute a SHA-256 Digest 416
135.1 Overview 416
135.2 Prerequisites 416
135.3 Syntax 416
135.4 Parameters 416
135.5 Return Value 416
135.6 Remarks 416
135.7 Example 416
135.8 API Reference 417
136 Compute a SHA3-256 Digest 417
136.1 Overview 417

136.2 Prerequisites 417
136.3 Syntax 417
136.4 Parameters 417
136.5 Return Value 417
136.6 Remarks 417
136.7 Example 418
136.8 API Reference 418
137 Compute a SHA3-512 Digest 418
137.1 Overview 418
137.2 Prerequisites 418
137.3 Syntax 418
137.4 Parameters 418
137.5 Return Value 418
137.6 Remarks 419
137.7 Example 419
137.8 API Reference 419
138 Compute a SHA-384 Digest 419
138.1 Overview 419
138.2 Prerequisites 419
138.3 Syntax 419
138.4 Parameters 419
138.5 Return Value 420
138.6 Remarks 420
138.7 Example 420
138.8 API Reference 420
139 Compute a SHA-512 Digest 420
139.1 Overview 420
139.2 Prerequisites 420
139.3 Syntax 420
139.4 Parameters 420
139.5 Return Value 421
139.6 Remarks 421

139.7 Example 421
139.8 API Reference 421
140 Generate a UUID v4 421
140.1 Overview 421
140.2 Prerequisites 421
140.3 Syntax 422
140.4 Parameters 422
140.5 Return Value 422
140.6 Remarks 422
140.7 Example 422
140.8 API Reference 422
141 Verify a Password Against a bcrypt Hash 422
141.1 Overview 422
141.2 Prerequisites 422
141.3 Syntax 423
141.4 Parameters 423
141.5 Return Value 423
141.6 Remarks 423
141.7 Example 423
141.8 API Reference 423
142 BCryptCost Property 424
142.1 Overview 424
142.2 Prerequisites 424
142.3 Syntax 424
142.4 Return Value 424
142.5 Remarks 424
142.6 API Reference 424
142.7 Code Example 424
143 CanReuseTransform Property 425
143.1 Overview 425
143.2 Prerequisites 425
143.3 Syntax 425

143.4 Return Value 425
143.5 Remarks 425
143.6 API Reference 425
143.7 Code Example 425
144 Hash Property 426
144.1 Overview 426
144.2 Prerequisites 426
144.3 Syntax 426
144.4 Return Value 426
144.5 Remarks 426
144.6 API Reference 426
144.7 Code Example 426
145 HashSize Property 427
145.1 Overview 427
145.2 Prerequisites 427
145.3 Syntax 427
145.4 Return Value 427
145.5 Remarks 427
145.6 API Reference 427
145.7 Code Example 427
146 Use the G3JSON Library 428
146.1 Overview 428
146.2 Prerequisites 428
146.3 How It Works 428
146.4 API Reference 428
146.5 Example 428
147 G3JSON Methods 429
147.1 Overview 429
147.2 Methods 429
147.3 Remarks 430
148 G3JSON Properties 430
148.1 Overview 430

148.2 Properties 430
148.3 Remarks 430
149 Load JSON from File 430
149.1 Overview 430
149.2 Prerequisites 430
149.3 Syntax 430
149.4 Parameters 430
149.5 Return Value 430
149.6 Remarks 431
149.7 Example 431
149.8 API Reference 431
150 Create a New JSON Array Container 431
150.1 Overview 431
150.2 Prerequisites 431
150.3 Syntax 431
150.4 Parameters 432
150.5 Return Value 432
150.6 Remarks 432
150.7 Example 432
150.8 API Reference 432
151 Create a New JSON Object Container 432
151.1 Overview 432
151.2 Prerequisites 432
151.3 Syntax 432
151.4 Parameters 432
151.5 Return Value 433
151.6 Remarks 433
151.7 Example 433
151.8 API Reference 433
152 Parse JSON Text 433
152.1 Overview 433
152.2 Prerequisites 433

152.3 Syntax 433
152.4 Parameters 433
152.5 Return Value 434
152.6 Remarks 434
152.7 Example 434
152.8 API Reference 434
153 Serialize Data to JSON Text 434
153.1 Overview 434
153.2 Prerequisites 434
153.3 Syntax 434
153.4 Parameters 435
153.5 Return Value 435
153.6 Remarks 435
153.7 Example 435
153.8 API Reference 435
154 Use the G3DB Library 436
154.1 Overview 436
154.2 Prerequisites 436
154.3 How It Works 436
154.4 API Reference 436
154.5 Example 437
155 G3DB Methods 437
155.1 Overview 437
155.2 Methods 438
155.3 Remarks 439
156 G3DB Properties 439
156.1 Overview 439
156.2 Properties 439
156.3 Remarks 440
157 Begin a Transaction 440
157.1 Overview 440
157.2 Prerequisites 440

157.3 Syntax 440
157.4 Parameters 440
157.5 Return Value 440
157.6 Remarks 440
157.7 Example 440
157.8 API Reference 441
158 Begin a Transaction with Options 441
158.1 Overview 441
158.2 Prerequisites 441
158.3 Syntax 441
158.4 Parameters 441
158.5 Return Value 441
158.6 Remarks 442
158.7 Example 442
158.8 API Reference 442
159 Close a Database Connection 442
159.1 Overview 442
159.2 Prerequisites 442
159.3 Syntax 442
159.4 Parameters 442
159.5 Return Value 442
159.6 Remarks 443
159.7 Example 443
159.8 API Reference 443
160 Execute a Non-Query SQL Statement 443
160.1 Overview 443
160.2 Prerequisites 443
160.3 Syntax 443
160.4 Parameters 443
160.5 Return Value 444
160.6 Remarks 444
160.7 Example 444

160.8 API Reference 444
161 Get the Current Error Message 444
161.1 Overview 444
161.2 Prerequisites 444
161.3 Syntax 444
161.4 Parameters 445
161.5 Return Value 445
161.6 Remarks 445
161.7 Example 445
161.8 API Reference 445
162 Open a Database Connection 445
162.1 Overview 445
162.2 Prerequisites 445
162.3 Syntax 445
162.4 Parameters 446
162.5 Return Value 446
162.6 Remarks 446
162.7 Example 446
162.8 API Reference 446
163 Open a Connection from Configuration 446
163.1 Overview 446
163.2 Prerequisites 447
163.3 Syntax 447
163.4 Parameters 447
163.5 Return Value 447
163.6 Remarks 447
163.7 Example 447
163.8 API Reference 447
164 Prepare a Reusable SQL Statement 448
164.1 Overview 448
164.2 Prerequisites 448
164.3 Syntax 448

164.4 Parameters 448
164.5 Return Value 448
164.6 Remarks 448
164.7 Example 448
164.8 API Reference 448
165 Execute a Query and Return a Result Set 449
165.1 Overview 449
165.2 Prerequisites 449
165.3 Syntax 449
165.4 Parameters 449
165.5 Return Value 449
165.6 Remarks 449
165.7 Example 449
165.8 API Reference 449
166 Execute a Query and Return One Row 450
166.1 Overview 450
166.2 Prerequisites 450
166.3 Syntax 450
166.4 Parameters 450
166.5 Return Value 450
166.6 Remarks 450
166.7 Example 450
166.8 API Reference 451
167 Set Maximum Connection Idle Time 451
167.1 Overview 451
167.2 Prerequisites 451
167.3 Syntax 451
167.4 Parameters 451
167.5 Return Value 451
167.6 Remarks 451
167.7 Example 451
167.8 API Reference 452

168 Set Maximum Connection Lifetime 452
168.1 Overview 452
168.2 Prerequisites 452
168.3 Syntax 452
168.4 Parameters 452
168.5 Return Value 452
168.6 Remarks 452
168.7 Example 452
168.8 API Reference 453
169 Set Maximum Idle Connections 453
169.1 Overview 453
169.2 Prerequisites 453
169.3 Syntax 453
169.4 Parameters 453
169.5 Return Value 453
169.6 Remarks 453
169.7 Example 453
169.8 API Reference 454
170 Set Maximum Open Connections 454
170.1 Overview 454
170.2 Prerequisites 454
170.3 Syntax 454
170.4 Parameters 454
170.5 Return Value 454
170.6 Remarks 454
170.7 Example 454
170.8 API Reference 455
171 Read Database Pool Statistics 455
171.1 Overview 455
171.2 Prerequisites 455
171.3 Syntax 455
171.4 Parameters 455

171.5 Return Value 455
171.6 Remarks 455
171.7 Example 455
171.8 API Reference 456
172 Driver Property 456
172.1 Overview 456
172.2 Prerequisites 456
172.3 Syntax 456
172.4 Return Value 456
172.5 Remarks 456
172.6 Example 456
172.7 API Reference 457
173 DSN Property 457
173.1 Overview 457
173.2 Prerequisites 457
173.3 Syntax 457
173.4 Return Value 457
173.5 Remarks 457
173.6 Example 457
173.7 API Reference 457
174 IsOpen Property 458
174.1 Overview 458
174.2 Prerequisites 458
174.3 Syntax 458
174.4 Return Value 458
174.5 Remarks 458
174.6 Example 458
174.7 API Reference 458
175 LastError Property 458
175.1 Overview 458
175.2 Prerequisites 458
175.3 Syntax 458

175.4 Return Value 459
175.5 Remarks 459
175.6 Example 459
175.7 API Reference 459
176 Use the G3HTTP Library 459
176.1 Overview 459
176.2 Prerequisites 459
176.3 How It Works 459
176.4 API Reference 460
176.5 Example 460
177 G3HTTP Methods 460
177.1 Overview 460
177.2 Methods 460
177.3 Remarks 461
178 G3HTTP Properties 461
178.1 Overview 461
178.2 Properties 461
178.3 Remarks 461
179 Send an HTTP Request with Fetch 461
179.1 Overview 461
179.2 Prerequisites 462
179.3 Syntax 462
179.4 Parameters 462
179.5 Return Value 462
179.6 Remarks 462
179.7 Example 462
179.8 API Reference 463
180 Use the G3MAIL Library 463
180.1 Overview 463
180.2 Syntax 463
180.3 Prerequisites 463
180.4 How it Works 464

180.5 API Reference 464
180.6 Code Example 464
181 G3MAIL Methods 465
181.1 Overview 465
181.2 Methods 465
181.3 Remarks 466
182 G3MAIL Properties 466
182.1 Overview 466
182.2 Properties 466
182.3 Remarks 467
183 AddAddress Method 467
183.1 Overview 467
183.2 Syntax 467
183.3 Parameters and Arguments 468
183.4 Return Values 468
183.5 Remarks 468
183.6 Code Example 468
184 AddBcc Method 468
184.1 Overview 468
184.2 Syntax 468
184.3 Parameters and Arguments 468
184.4 Return Values 469
184.5 Remarks 469
184.6 Code Example 469
185 AddCc Method 469
185.1 Overview 469
185.2 Syntax 469
185.3 Parameters and Arguments 469
185.4 Return Values 469
185.5 Remarks 469
185.6 Code Example 470
186 Clear Method 470

186.1 Overview 470
186.2 Syntax 470
186.3 Parameters and Arguments 470
186.4 Return Values 470
186.5 Remarks 470
186.6 Code Example 470
187 Send Method 471
187.1 Overview 471
187.2 Syntax 471
187.3 Parameters and Arguments 471
187.4 Return Values 471
187.5 Remarks 471
187.6 Code Example 472
188 Bcc Property 472
188.1 Overview 472
188.2 Syntax 472
188.3 Parameters and Arguments 472
188.4 Return Values 472
188.5 Remarks 472
188.6 Code Example 473
189 Body Property 473
189.1 Overview 473
189.2 Syntax 473
189.3 Parameters and Arguments 473
189.4 Return Values 473
189.5 Remarks 473
189.6 Code Example 473
190 BodyFormat Property 474
190.1 Overview 474
190.2 Syntax 474
190.3 Parameters and Arguments 474
190.4 Return Values 474

190.5 Remarks 474
190.6 Code Example 474
191 Cc Property 475
191.1 Overview 475
191.2 Syntax 475
191.3 Parameters and Arguments 475
191.4 Return Values 475
191.5 Remarks 475
191.6 Code Example 475
192 From Property 475
192.1 Overview 475
192.2 Syntax 475
192.3 Parameters and Arguments 476
192.4 Return Values 476
192.5 Remarks 476
192.6 Code Example 476
193 FromName Property 476
193.1 Overview 476
193.2 Syntax 476
193.3 Parameters and Arguments 476
193.4 Return Values 476
193.5 Remarks 477
193.6 Code Example 477
194 Host Property 477
194.1 Overview 477
194.2 Syntax 477
194.3 Parameters and Arguments 477
194.4 Return Values 477
194.5 Remarks 477
194.6 Code Example 477
195 IsHTML Property 478
195.1 Overview 478

195.2 Syntax 478
195.3 Parameters and Arguments 478
195.4 Return Values 478
195.5 Remarks 478
195.6 Code Example 478
196 Password Property 478
196.1 Overview 478
196.2 Syntax 479
196.3 Parameters and Arguments 479
196.4 Return Values 479
196.5 Remarks 479
196.6 Code Example 479
197 Port Property 479
197.1 Overview 479
197.2 Syntax 479
197.3 Parameters and Arguments 479
197.4 Return Values 479
197.5 Remarks 480
197.6 Code Example 480
198 Subject Property 480
198.1 Overview 480
198.2 Syntax 480
198.3 Parameters and Arguments 480
198.4 Return Values 480
198.5 Remarks 480
198.6 Code Example 480
199 To Property 481
199.1 Overview 481
199.2 Syntax 481
199.3 Parameters and Arguments 481
199.4 Return Values 481
199.5 Remarks 481

199.6 Code Example 481
200 Username Property 481
200.1 Overview 481
200.2 Syntax 482
200.3 Parameters and Arguments 482
200.4 Return Values 482
200.5 Remarks 482
200.6 Code Example 482
201 Use the G3IMAGE Library 482
201.1 Overview 482
201.2 Syntax 482
201.3 Prerequisites 483
201.4 How it Works 483
201.5 API Reference 483
201.6 Code Example 484
202 G3IMAGE Methods 485
202.1 Overview 485
202.2 Methods 485
202.3 Remarks 487
203 G3IMAGE Properties 487
203.1 Overview 487
203.2 Properties 487
203.3 Remarks 489
204 Clear Method 489
204.1 Overview 489
204.2 Syntax 489
204.3 Return Values 489
204.4 Remarks 489
204.5 Code Example 489
205 Close Method 490
205.1 Overview 490
205.2 Syntax 490

205.3 Return Values 490
205.4 Remarks 490
205.5 Code Example 490
206 DrawCircle Method 490
206.1 Overview 490
206.2 Syntax 490
206.3 Parameters 490
206.4 Return Values 490
206.5 Remarks 491
206.6 Code Example 491
207 DrawEllipse Method 491
207.1 Overview 491
207.2 Syntax 491
207.3 Parameters 491
207.4 Return Values 491
207.5 Remarks 491
207.6 Code Example 491
208 DrawImage Method 492
208.1 Overview 492
208.2 Syntax 492
208.3 Parameters 492
208.4 Return Values 492
208.5 Remarks 492
208.6 Code Example 492
209 DrawLine Method 493
209.1 Overview 493
209.2 Syntax 493
209.3 Parameters 493
209.4 Return Values 493
209.5 Remarks 493
209.6 Code Example 493
210 DrawRectangle Method 493

210.1 Overview 493
210.2 Syntax 493
210.3 Parameters 494
210.4 Return Values 494
210.5 Remarks 494
210.6 Code Example 494
211 DrawString Method 494
211.1 Overview 494
211.2 Syntax 494
211.3 Parameters 494
211.4 Return Values 494
211.5 Remarks 495
211.6 Code Example 495
212 DrawStringAnchored Method 495
212.1 Overview 495
212.2 Syntax 495
212.3 Parameters 495
212.4 Return Values 495
212.5 Remarks 495
212.6 Code Example 496
213 Fill Method 496
213.1 Overview 496
213.2 Syntax 496
213.3 Return Values 496
213.4 Remarks 496
213.5 Code Example 496
214 LoadFontFace Method 497
214.1 Overview 497
214.2 Syntax 497
214.3 Parameters 497
214.4 Return Values 497
214.5 Remarks 497

214.6 Code Example 497
215 LoadImage Method 497
215.1 Overview 497
215.2 Syntax 497
215.3 Parameters 497
215.4 Return Values 498
215.5 Remarks 498
215.6 Code Example 498
216 LoadJPG Method 498
216.1 Overview 498
216.2 Syntax 498
216.3 Parameters 498
216.4 Return Values 498
216.5 Remarks 498
216.6 Code Example 498
217 LoadPNG Method 499
217.1 Overview 499
217.2 Syntax 499
217.3 Parameters 499
217.4 Return Values 499
217.5 Remarks 499
217.6 Code Example 499
218 MeasureString Method 499
218.1 Overview 499
218.2 Syntax 500
218.3 Parameters 500
218.4 Return Values 500
218.5 Remarks 500
218.6 Code Example 500
219 NewContext Method 500
219.1 Overview 500
219.2 Syntax 500

219.3 Parameters 500
219.4 Return Values 500
219.5 Remarks 501
219.6 Code Example 501
220 RenderViaTemp Method 501
220.1 Overview 501
220.2 Syntax 501
220.3 Parameters 501
220.4 Return Values 501
220.5 Remarks 501
220.6 Code Example 501
221 SaveJPG Method 502
221.1 Overview 502
221.2 Syntax 502
221.3 Parameters 502
221.4 Return Values 502
221.5 Remarks 502
221.6 Code Example 502
222 SavePNG Method 503
222.1 Overview 503
222.2 Syntax 503
222.3 Parameters 503
222.4 Return Values 503
222.5 Remarks 503
222.6 Code Example 503
223 SetColor Method 503
223.1 Overview 503
223.2 Syntax 503
223.3 Parameters 504
223.4 Return Values 504
223.5 Remarks 504
223.6 Code Example 504

224 SetHexColor Method 504
224.1 Overview 504
224.2 Syntax 504
224.3 Parameters 504
224.4 Return Values 504
224.5 Remarks 504
224.6 Code Example 505
225 SetLineWidth Method 505
225.1 Overview 505
225.2 Syntax 505
225.3 Parameters 505
225.4 Return Values 505
225.5 Remarks 505
225.6 Code Example 505
226 Stroke Method 506
226.1 Overview 506
226.2 Syntax 506
226.3 Return Values 506
226.4 Remarks 506
226.5 Code Example 506
227 DefaultFormat Property 506
227.1 Overview 506
227.2 Syntax 506
227.3 Return Values 507
227.4 Remarks 507
227.5 Code Example 507
228 HasContext Property 507
228.1 Overview 507
228.2 Syntax 507
228.3 Return Values 507
228.4 Remarks 507
228.5 Code Example 507

229 Height Property 508
229.1 Overview 508
229.2 Syntax 508
229.3 Return Values 508
229.4 Remarks 508
229.5 Code Example 508
230 JpgQuality Property 508
230.1 Overview 508
230.2 Syntax 508
230.3 Return Values 508
230.4 Remarks 508
230.5 Code Example 509
231 LastBytes Property 509
231.1 Overview 509
231.2 Syntax 509
231.3 Return Values 509
231.4 Remarks 509
231.5 Code Example 509
232 LastError Property 509
232.1 Overview 509
232.2 Syntax 510
232.3 Return Values 510
232.4 Remarks 510
232.5 Code Example 510
233 LastMimeType Property 510
233.1 Overview 510
233.2 Syntax 510
233.3 Return Values 510
233.4 Remarks 510
233.5 Code Example 510
234 LastTempFile Property 511
234.1 Overview 511

234.2 Syntax 511
234.3 Return Values 511
234.4 Remarks 511
234.5 Code Example 511
235 Width Property 511
235.1 Overview 511
235.2 Syntax 511
235.3 Return Values 512
235.4 Remarks 512
235.5 Code Example 512
236 Use the G3FILES Library 512
236.1 Overview 512
236.2 Syntax 512
236.3 Prerequisites 512
236.4 How it Works 512
236.5 API Reference 513
236.6 Code Example 513
237 G3FILES Methods 514
237.1 Overview 514
237.2 Methods 514
237.3 Remarks 515
238 G3FILES Properties 516
238.1 Overview 516
238.2 Properties 516
238.3 Remarks 516
239 Append Method 516
239.1 Overview 516
239.2 Syntax 516
239.3 Parameters and Arguments 516
239.4 Return Values 516
239.5 Remarks 517
239.6 Code Example 517

240 ConvertFileEncoding Method 517
240.1 Overview 517
240.2 Syntax 517
240.3 Parameters and Arguments 517
240.4 Return Values 517
240.5 Remarks 518
240.6 Code Example 518
241 ConvertTextEncoding Method 518
241.1 Overview 518
241.2 Syntax 518
241.3 Parameters and Arguments 518
241.4 Return Values 518
241.5 Remarks 518
241.6 Code Example 518
242 Copy Method 519
242.1 Overview 519
242.2 Syntax 519
242.3 Parameters and Arguments 519
242.4 Return Values 519
242.5 Remarks 519
242.6 Code Example 519
243 Delete Method 519
243.1 Overview 519
243.2 Syntax 519
243.3 Parameters and Arguments 520
243.4 Return Values 520
243.5 Remarks 520
243.6 Code Example 520
244 Exists Method 520
244.1 Overview 520
244.2 Syntax 520
244.3 Parameters and Arguments 520

244.4 Return Values 520
244.5 Remarks 520
244.6 Code Example 521
245 List Method 521
245.1 Overview 521
245.2 Syntax 521
245.3 Parameters and Arguments 521
245.4 Return Values 521
245.5 Remarks 521
245.6 Code Example 521
246 Mkdir Method 522
246.1 Overview 522
246.2 Syntax 522
246.3 Parameters and Arguments 522
246.4 Return Values 522
246.5 Remarks 522
246.6 Code Example 522
247 Move Method 522
247.1 Overview 522
247.2 Syntax 522
247.3 Parameters and Arguments 523
247.4 Return Values 523
247.5 Remarks 523
247.6 Code Example 523
248 NormalizeEOL Method 523
248.1 Overview 523
248.2 Syntax 523
248.3 Parameters and Arguments 523
248.4 Return Values 523
248.5 Remarks 523
248.6 Code Example 524
249 Read Method 524

249.1 Overview 524
249.2 Syntax 524
249.3 Parameters and Arguments 524
249.4 Return Values 524
249.5 Remarks 524
249.6 Code Example 524
250 Size Method 525
250.1 Overview 525
250.2 Syntax 525
250.3 Parameters and Arguments 525
250.4 Return Values 525
250.5 Remarks 525
250.6 Code Example 525
251 Write Method 525
251.1 Overview 525
251.2 Syntax 525
251.3 Parameters and Arguments 525
251.4 Return Values 526
251.5 Remarks 526
251.6 Code Example 526
252 Use the G3TESTSUITE Library 526
252.1 Overview 526
252.2 Syntax 526
252.3 Prerequisites 527
252.4 How it Works 527
252.5 API Reference 527
252.6 Code Example 528
253 G3TESTSUITE Methods 528
253.1 Overview 528
253.2 Methods 528
253.3 Remarks 529
254 G3TESTSUITE Properties 529

254.1 Overview 529
254.2 Properties 530
254.3 Remarks 530
255 Use the G3TEMPLATE Library 530
255.1 Overview 530
255.2 Syntax 530
255.3 Prerequisites 531
255.4 How it Works 531
255.5 API Reference 531
255.6 Code Example 531
256 G3TEMPLATE Methods 532
256.1 Overview 532
256.2 Methods 532
256.3 Remarks 532
257 G3TEMPLATE Properties 532
257.1 Overview 532
257.2 Properties 532
257.3 Remarks 532
258 Render Method 532
258.1 Overview 532
258.2 Syntax 533
258.3 Parameters and Arguments 533
258.4 Return Values 533
258.5 Remarks 533
258.6 Code Example 533
259 Use the G3ZIP Library 534
259.1 Overview 534
259.2 Syntax 534
259.3 Prerequisites 534
259.4 How it Works 534
259.5 API Reference 534
259.6 Code Example 535

260 G3ZIP Methods 535
260.1 Overview 535
260.2 Methods 535
260.3 Remarks 537
261 G3ZIP Properties 537
261.1 Overview 537
261.2 Properties 537
261.3 Remarks 537
262 AddFile Method 537
262.1 Overview 537
262.2 Syntax 537
262.3 Parameters and Arguments 537
262.4 Return Values 538
262.5 Remarks 538
262.6 Code Example 538
263 AddFolder Method 538
263.1 Overview 538
263.2 Syntax 538
263.3 Parameters and Arguments 538
263.4 Return Values 538
263.5 Remarks 538
263.6 Code Example 539
264 AddText Method 539
264.1 Overview 539
264.2 Syntax 539
264.3 Parameters and Arguments 539
264.4 Return Values 539
264.5 Remarks 539
264.6 Code Example 539
265 Close Method 540
265.1 Overview 540
265.2 Syntax 540

265.3 Return Values 540
265.4 Remarks 540
265.5 Code Example 540
266 Create Method 540
266.1 Overview 540
266.2 Syntax 540
266.3 Parameters and Arguments 541
266.4 Return Values 541
266.5 Remarks 541
266.6 Code Example 541
267 ExtractAll Method 541
267.1 Overview 541
267.2 Syntax 541
267.3 Parameters and Arguments 541
267.4 Return Values 541
267.5 Remarks 542
267.6 Code Example 542
268 ExtractFile Method 542
268.1 Overview 542
268.2 Syntax 542
268.3 Parameters and Arguments 542
268.4 Return Values 542
268.5 Remarks 542
268.6 Code Example 543
269 GetInfo Method 543
269.1 Overview 543
269.2 Syntax 543
269.3 Parameters and Arguments 543
269.4 Return Values 543
269.5 Remarks 543
269.6 Code Example 543
270 List Method 544

270.1 Overview 544
270.2 Syntax 544
270.3 Return Values 544
270.4 Remarks 544
270.5 Code Example 544
271 Open Method 545
271.1 Overview 545
271.2 Syntax 545
271.3 Parameters and Arguments 545
271.4 Return Values 545
271.5 Remarks 545
271.6 Code Example 545
272 Count Property 545
272.1 Overview 545
272.2 Syntax 545
272.3 Return Values 546
272.4 Remarks 546
272.5 Code Example 546
273 Mode Property 546
273.1 Overview 546
273.2 Syntax 546
273.3 Return Values 546
273.4 Remarks 546
273.5 Code Example 546
274 Path Property 547
274.1 Overview 547
274.2 Syntax 547
274.3 Return Values 547
274.4 Remarks 547
274.5 Code Example 547
275 Use G3ZLIB in G3Pix AxonASP 547
275.1 Overview 547

275.2 Syntax 547
275.3 Parameters and Arguments 548
275.4 Return Values 548
275.5 Remarks 548
275.6 Code Example 548
276 G3ZLIB Methods 548
276.1 Overview 548
276.2 Methods 548
276.3 Remarks 549
277 G3ZLIB Properties 549
277.1 Overview 549
277.2 Properties 549
277.3 Remarks 550
278 Clear G3ZLIB State 550
278.1 Overview 550
278.2 Syntax 550
278.3 Parameters and Arguments 550
278.4 Return Values 550
278.5 Remarks 550
278.6 Code Example 550
279 Compress Data Using G3ZLIB 551
279.1 Overview 551
279.2 Syntax 551
279.3 Parameters and Arguments 551
279.4 Return Values 551
279.5 Remarks 551
279.6 Code Example 551
280 Compress a File Using G3ZLIB 551
280.1 Overview 551
280.2 Syntax 552
280.3 Parameters and Arguments 552
280.4 Return Values 552

280.5 Remarks 552
280.6 Code Example 552
281 Compress Multiple Files Using G3ZLIB 552
281.1 Overview 552
281.2 Syntax 553
281.3 Parameters and Arguments 553
281.4 Return Values 553
281.5 Remarks 553
281.6 Code Example 553
282 Decompress Data Using G3ZLIB 554
282.1 Overview 554
282.2 Syntax 554
282.3 Parameters and Arguments 554
282.4 Return Values 554
282.5 Remarks 554
282.6 Code Example 554
283 Decompress a File Using G3ZLIB 554
283.1 Overview 554
283.2 Syntax 554
283.3 Parameters and Arguments 555
283.4 Return Values 555
283.5 Remarks 555
283.6 Code Example 555
284 Decompress Multiple Files Using G3ZLIB 555
284.1 Overview 555
284.2 Syntax 555
284.3 Parameters and Arguments 555
284.4 Return Values 556
284.5 Remarks 556
284.6 Code Example 556
285 Decompress Text Using G3ZLIB 556
285.1 Overview 556

285.2 Syntax 556
285.3 Parameters and Arguments 556
285.4 Return Values 557
285.5 Remarks 557
285.6 Code Example 557
286 Get the LastError Property 557
286.1 Overview 557
286.2 Syntax 557
286.3 Parameters and Arguments 557
286.4 Return Values 557
286.5 Remarks 558
286.6 Code Example 558
287 Use G3TAR in AxonASP 558
287.1 Overview 558
287.2 Syntax 558
287.3 Parameters and Arguments 558
287.4 Return Values 558
287.5 Remarks 558
287.6 Code Example 559
288 G3TAR Methods 559
288.1 Overview 559
288.2 Methods 559
288.3 Remarks 560
289 G3TAR Properties 560
289.1 Overview 560
289.2 Properties 560
289.3 Remarks 561
290 AddFile Method 561
290.1 Overview 561
290.2 Syntax 561
290.3 Parameters and Arguments 561
290.4 Return Values 561

290.5 Remarks 561
290.6 Code Example 561
291 AddFiles Method 562
291.1 Overview 562
291.2 Syntax 562
291.3 Parameters and Arguments 562
291.4 Return Values 562
291.5 Remarks 562
291.6 Code Example 562
292 AddFolder Method 563
292.1 Overview 563
292.2 Syntax 563
292.3 Parameters and Arguments 563
292.4 Return Values 563
292.5 Remarks 563
292.6 Code Example 563
293 AddText Method 564
293.1 Overview 564
293.2 Syntax 564
293.3 Parameters and Arguments 564
293.4 Return Values 564
293.5 Remarks 564
293.6 Code Example 564
294 Close Method 565
294.1 Overview 565
294.2 Syntax 565
294.3 Parameters and Arguments 565
294.4 Return Values 565
294.5 Remarks 565
294.6 Code Example 565
295 Create Method 565
295.1 Overview 565

295.2 Syntax 566
295.3 Parameters and Arguments 566
295.4 Return Values 566
295.5 Remarks 566
295.6 Code Example 566
296 ExtractAll Method 566
296.1 Overview 566
296.2 Syntax 566
296.3 Parameters and Arguments 567
296.4 Return Values 567
296.5 Remarks 567
296.6 Code Example 567
297 ExtractFile Method 567
297.1 Overview 567
297.2 Syntax 567
297.3 Parameters and Arguments 567
297.4 Return Values 568
297.5 Remarks 568
297.6 Code Example 568
298 GetInfo Method 568
298.1 Overview 568
298.2 Syntax 568
298.3 Parameters and Arguments 568
298.4 Return Values 569
298.5 Remarks 569
298.6 Code Example 569
299 List Method 569
299.1 Overview 569
299.2 Syntax 569
299.3 Parameters and Arguments 569
299.4 Return Values 569
299.5 Remarks 570

299.6 Code Example 570
300 Open Method 570
300.1 Overview 570
300.2 Syntax 570
300.3 Parameters and Arguments 570
300.4 Return Values 570
300.5 Remarks 570
300.6 Code Example 571
301 Count Property 571
301.1 Overview 571
301.2 Syntax 571
301.3 Parameters and Arguments 571
301.4 Return Values 571
301.5 Remarks 571
301.6 Code Example 571
302 LastError Property 572
302.1 Overview 572
302.2 Syntax 572
302.3 Parameters and Arguments 572
302.4 Return Values 572
302.5 Remarks 572
302.6 Code Example 572
303 Mode Property 572
303.1 Overview 572
303.2 Syntax 572
303.3 Parameters and Arguments 572
303.4 Return Values 573
303.5 Remarks 573
303.6 Code Example 573
304 Path Property 573
304.1 Overview 573
304.2 Syntax 573

304.3 Parameters and Arguments 573
304.4 Return Values 573
304.5 Remarks 573
304.6 Code Example 573
305 Use the G3ZSTD Library 574
305.1 Overview 574
305.2 Syntax 574
305.3 Prerequisites 574
305.4 How it Works 574
305.5 API Reference 574
305.6 Code Example 575
306 G3ZSTD Methods 576
306.1 Overview 576
306.2 Methods Reference 576
306.3 Remarks 577
307 G3ZSTD Properties 577
307.1 Overview 577
307.2 Properties Reference 577
307.3 Remarks 577
308 Clear Method 577
308.1 Overview 577
308.2 Syntax 577
308.3 Parameters and Arguments 578
308.4 Return Values 578
308.5 Remarks 578
308.6 Code Example 578
309 Compress Method 578
309.1 Overview 578
309.2 Syntax 578
309.3 Parameters and Arguments 578
309.4 Return Values 579
309.5 Remarks 579

309.6 Code Example 579
310 CompressFile Method 579
310.1 Overview 579
310.2 Syntax 579
310.3 Parameters and Arguments 580
310.4 Return Values 580
310.5 Remarks 580
310.6 Code Example 580
311 CompressMany Method 580
311.1 Overview 580
311.2 Syntax 581
311.3 Parameters and Arguments 581
311.4 Return Values 581
311.5 Remarks 581
311.6 Code Example 581
312 Decompress Method 582
312.1 Overview 582
312.2 Syntax 582
312.3 Parameters and Arguments 582
312.4 Return Values 582
312.5 Remarks 582
312.6 Code Example 582
313 DecompressFile Method 583
313.1 Overview 583
313.2 Syntax 583
313.3 Parameters and Arguments 583
313.4 Return Values 583
313.5 Remarks 583
313.6 Code Example 583
314 DecompressMany Method 584
314.1 Overview 584
314.2 Syntax 584

314.3 Parameters and Arguments 584
314.4 Return Values 584
314.5 Remarks 584
314.6 Code Example 584
315 DecompressText Method 585
315.1 Overview 585
315.2 Syntax 585
315.3 Parameters and Arguments 585
315.4 Return Values 585
315.5 Remarks 585
315.6 Code Example 585
316 SetLevel Method 586
316.1 Overview 586
316.2 Syntax 586
316.3 Parameters and Arguments 586
316.4 Return Values 586
316.5 Remarks 586
316.6 Code Example 586
317 LastError Property 587
317.1 Overview 587
317.2 Syntax 587
317.3 Parameters and Arguments 587
317.4 Return Values 587
317.5 Remarks 587
317.6 Code Example 587
318 Level Property 587
318.1 Overview 587
318.2 Syntax 588
318.3 Parameters and Arguments 588
318.4 Return Values 588
318.5 Remarks 588
318.6 Code Example 588

319 Use the G3FC Archive Library in G3Pix AxonASP 588
319.1 Overview 588
319.2 Syntax 588
319.3 Parameters and Arguments 589
319.4 Return Values 589
319.5 Remarks 589
319.6 Code Example 589
319.7 Methods Reference 590
319.8 Properties Reference 591
319.9 API Reference 591
320 G3FC Methods 591
320.1 Overview 591
320.2 Methods Reference 591
320.3 Remarks 593
321 G3FC Properties 593
321.1 Overview 593
321.2 Properties Reference 593
321.3 Remarks 593
322 Create Method 593
322.1 Overview 593
322.2 Syntax 593
322.3 Parameters and Arguments 593
322.4 Return Values 594
322.5 Remarks 594
322.6 Code Example 594
323 Extract Method 595
323.1 Overview 595
323.2 Syntax 595
323.3 Parameters and Arguments 595
323.4 Return Values 595
323.5 Remarks 595
323.6 Code Example 595

324 ExtractSingle Method 596
324.1 Overview 596
324.2 Syntax 596
324.3 Parameters and Arguments 596
324.4 Return Values 596
324.5 Remarks 596
324.6 Code Example 596
325 Find Method 597
325.1 Overview 597
325.2 Syntax 597
325.3 Parameters and Arguments 597
325.4 Return Values 597
325.5 Remarks 597
325.6 Code Example 597
326 Info Method 598
326.1 Overview 598
326.2 Syntax 598
326.3 Parameters and Arguments 598
326.4 Return Values 598
326.5 Remarks 598
326.6 Code Example 598
327 List Method 599
327.1 Overview 599
327.2 Syntax 599
327.3 Parameters and Arguments 599
327.4 Return Values 599
327.5 Remarks 599
327.6 Code Example 600
328 Use the G3MD Library 600
328.1 Overview 600
328.2 Syntax 600
328.3 Prerequisites 600

328.4 How it Works 601
328.5 API Reference 601
328.6 Code Example 601
329 G3MD Methods 602
329.1 Overview 602
329.2 Methods Reference 602
329.3 Remarks 602
330 G3MD Properties 602
330.1 Overview 602
330.2 Properties Reference 602
330.3 Remarks 602
331 Process Method 603
331.1 Overview 603
331.2 Syntax 603
331.3 Parameters and Arguments 603
331.4 Return Values 603
331.5 Remarks 603
331.6 Code Example 603
332 HardWraps Property 604
332.1 Overview 604
332.2 Syntax 604
332.3 Return Values 604
332.4 Remarks 604
332.5 Code Example 604
333 Unsafe Property 605
333.1 Overview 605
333.2 Syntax 605
333.3 Return Values 605
333.4 Remarks 605
333.5 Code Example 605
334 Create PDF documents using G3PDF 606
334.1 Overview 606

334.2 Syntax 606
334.3 Parameters 606
334.4 Return Values 606
334.5 Remarks 606
334.6 Code Example 606
335 G3PDF Methods 606
335.1 Overview 606
335.2 Methods Reference 607
335.3 Remarks 610
336 G3PDF Properties 610
336.1 Overview 610
336.2 Properties Reference 610
336.3 Remarks 611
337 AddPage 611
337.1 Overview 611
337.2 Syntax 611
337.3 Parameters 611
337.4 Return Value 612
337.5 Code Example 612
338 Cell 612
338.1 Overview 612
338.2 Syntax 612
338.3 Parameters 612
338.4 Return Value 612
338.5 Code Example 613
339 Close 613
339.1 Overview 613
339.2 Syntax 613
339.3 Parameters 613
339.4 Return Value 613
339.5 Code Example 613
340 Image 614

340.1 Overview 614
340.2 Syntax 614
340.3 Parameters 614
340.4 Return Value 614
340.5 Code Example 614
341 Line 614
341.1 Overview 614
341.2 Syntax 615
341.3 Parameters 615
341.4 Return Value 615
341.5 Code Example 615
342 Ln 615
342.1 Overview 615
342.2 Syntax 615
342.3 Parameters 615
342.4 Return Value 615
342.5 Code Example 616
343 MultiCell 616
343.1 Overview 616
343.2 Syntax 616
343.3 Parameters 616
343.4 Return Value 616
343.5 Code Example 616
344 Rect 617
344.1 Overview 617
344.2 Syntax 617
344.3 Parameters 617
344.4 Return Value 617
344.5 Code Example 617
345 Reset 618
345.1 Overview 618
345.2 Syntax 618

345.3 Parameters 618
345.4 Return Value 618
345.5 Code Example 618
346 SetAutoPageBreak 618
346.1 Overview 618
346.2 Syntax 618
346.3 Parameters 618
346.4 Return Value 619
346.5 Code Example 619
347 SetDrawColor 619
347.1 Overview 619
347.2 Syntax 619
347.3 Parameters 619
347.4 Return Value 619
347.5 Code Example 619
348 SetFillColor 620
348.1 Overview 620
348.2 Syntax 620
348.3 Parameters 620
348.4 Return Value 620
348.5 Code Example 620
349 SetFont 621
349.1 Overview 621
349.2 Syntax 621
349.3 Parameters 621
349.4 Return Value 621
349.5 Code Example 621
350 SetFontSize 621
350.1 Overview 621
350.2 Syntax 621
350.3 Parameters 621
350.4 Return Value 622

350.5 Code Example 622
351 SetLeftMargin 622
351.1 Overview 622
351.2 Syntax 622
351.3 Parameters 622
351.4 Return Value 622
351.5 Code Example 622
352 SetLineWidth 623
352.1 Overview 623
352.2 Syntax 623
352.3 Parameters 623
352.4 Return Value 623
352.5 Code Example 623
353 SetMargins 623
353.1 Overview 623
353.2 Syntax 623
353.3 Parameters 623
353.4 Return Value 624
353.5 Code Example 624
354 SetRightMargin 624
354.1 Overview 624
354.2 Syntax 624
354.3 Parameters 624
354.4 Return Value 624
354.5 Code Example 624
355 SetTextColor 625
355.1 Overview 625
355.2 Syntax 625
355.3 Parameters 625
355.4 Return Value 625
355.5 Code Example 625
356 SetTopMargin 625

356.1 Overview 625
356.2 Syntax 626
356.3 Parameters 626
356.4 Return Value 626
356.5 Code Example 626
357 SetX 626
357.1 Overview 626
357.2 Syntax 626
357.3 Parameters 626
357.4 Return Value 626
357.5 Code Example 626
358 SetXY 627
358.1 Overview 627
358.2 Syntax 627
358.3 Parameters 627
358.4 Return Value 627
358.5 Code Example 627
359 SetY 628
359.1 Overview 628
359.2 Syntax 628
359.3 Parameters 628
359.4 Return Value 628
359.5 Code Example 628
360 Text 628
360.1 Overview 628
360.2 Syntax 628
360.3 Parameters 628
360.4 Return Value 629
360.5 Code Example 629
361 Write 629
361.1 Overview 629
361.2 Syntax 629

361.3 Parameters 629
361.4 Return Value 629
361.5 Code Example 629
362 H 630
362.1 Overview 630
362.2 Property Type 630
362.3 Code Example 630
363 LastError 630
363.1 Overview 630
363.2 Property Type 630
363.3 Code Example 631
364 Page 631
364.1 Overview 631
364.2 Property Type 631
364.3 Code Example 631
365 Version 631
365.1 Overview 631
365.2 Property Type 632
365.3 Code Example 632
366 W 632
366.1 Overview 632
366.2 Property Type 632
366.3 Code Example 632
367 X 633
367.1 Overview 633
367.2 Property Type 633
367.3 Code Example 633
368 Y 633
368.1 Overview 633
368.2 Property Type 633
368.3 Code Example 633
369 G3FILEUPLOADER Library 634

369.1 Overview 634
369.2 Syntax 634
369.3 Properties and Methods 634
369.4 Return Values 634
369.5 Example 634
370 G3FILEUPLOADER Methods 635
370.1 Overview 635
370.2 Methods Reference 635
370.3 Remarks 636
371 G3FILEUPLOADER Properties 636
371.1 Overview 636
371.2 Properties Reference 636
371.3 Remarks 637
372 AllowExtension Method 637
372.1 Overview 637
372.2 Syntax 637
372.3 Parameters and Arguments 637
372.4 Return Values 637
373 AllowExtensions Method 637
373.1 Overview 637
373.2 Syntax 637
373.3 Parameters and Arguments 637
373.4 Return Values 638
374 BlockExtension Method 638
374.1 Overview 638
374.2 Syntax 638
374.3 Parameters and Arguments 638
374.4 Return Values 638
375 BlockExtensions Method 638
375.1 Overview 638
375.2 Syntax 638
375.3 Parameters and Arguments 638

375.4 Return Values 638
376 Form Method 639
376.1 Overview 639
376.2 Syntax 639
376.3 Parameters and Arguments 639
376.4 Return Values 639
377 GetAllFilesInfo Method 639
377.1 Overview 639
377.2 Syntax 639
377.3 Return Values 639
378 GetFileInfo Method 640
378.1 Overview 640
378.2 Syntax 640
378.3 Parameters and Arguments 640
378.4 Return Values 640
379 IsValidExtension Method 640
379.1 Overview 640
379.2 Syntax 640
379.3 Parameters and Arguments 640
379.4 Return Values 640
380 Process Method 641
380.1 Overview 641
380.2 Syntax 641
380.3 Parameters and Arguments 641
380.4 Return Values 641
381 ProcessAll Method 641
381.1 Overview 641
381.2 Syntax 641
381.3 Parameters and Arguments 641
381.4 Return Values 642
382 SetUseAllowedOnly Method 642
382.1 Overview 642

382.2 Syntax 642
382.3 Parameters and Arguments 642
382.4 Return Values 642
383 AllowedExtensions Property 642
383.1 Overview 642
383.2 Syntax 642
383.3 Return Values 642
384 BlockedExtensions Property 643
384.1 Overview 643
384.2 Syntax 643
384.3 Return Values 643
385 DebugMode Property 643
385.1 Overview 643
385.2 Syntax 643
385.3 Return Values 643
386 MaxFileSize Property 643
386.1 Overview 643
386.2 Syntax 643
386.3 Return Values 643
387 PreserveOriginalName Property 644
387.1 Overview 644
387.2 Syntax 644
387.3 Return Values 644
388 Use WScript.Shell in AxonASP 644
388.1 Overview 644
388.2 Syntax 644
388.3 Parameters and Arguments 644
388.4 Return Values 644
388.5 Remarks 644
388.6 Code Example 644
389 WScript.Shell Methods 645
389.1 Overview 645

389.2 Methods Reference 645
389.3 Remarks 646
390 WScript.Shell Properties 646
390.1 Overview 646
390.2 Properties Reference 646
390.3 Remarks 646
391 Use CreateObject from WScript.Shell 646
391.1 Overview 646
391.2 Syntax 646
391.3 Parameters 647
391.4 Return Value 647
391.5 How It Works 647
391.6 Remarks 647
391.7 Example 647
392 Run a Command with Exec 647
392.1 Overview 647
392.2 Syntax 648
392.3 Parameters 648
392.4 Return Value 648
392.5 How It Works 648
392.6 Remarks 648
392.7 Example 648
393 Expand Environment Variables in Text 649
393.1 Overview 649
393.2 Syntax 649
393.3 Parameters 649
393.4 Return Value 649
393.5 How It Works 649
393.6 Remarks 649
393.7 Example 649
394 Read an Environment Variable with GetEnv 650
394.1 Overview 650

394.2 Syntax 650
394.3 Parameters 650
394.4 Return Value 650
394.5 How It Works 650
394.6 Remarks 650
394.7 Example 650
395 Run a Command with Run 651
395.1 Overview 651
395.2 Syntax 651
395.3 Parameters 651
395.4 Return Value 651
395.5 How It Works 651
395.6 Remarks 651
395.7 Example 651
396 Use ADOX.Catalog in G3Pix AxonASP 652
396.1 Overview 652
396.2 Prerequisites 652
396.3 Syntax 652
396.4 Parameters and Arguments 652
396.5 Return Value 652
396.6 How It Works 652
396.7 Remarks 652
396.8 Example 653
397 ADOX.Catalog Methods 653
397.1 Overview 653
397.2 Methods Reference 653
397.3 Remarks 653
398 ADOX.Catalog Properties 653
398.1 Overview 653
398.2 Properties Reference 654
398.3 Remarks 654
399 Set Catalog ActiveConnection 654

399.1 Overview 654
399.2 Syntax 654
399.3 Parameters 654
399.4 Return Value 654
399.5 How It Works 655
399.6 Remarks 655
399.7 Example 655
400 Get Catalog Tables 655
400.1 Overview 655
400.2 Prerequisites 655
400.3 Syntax 655
400.4 Parameters 656
400.5 Return Value 656
400.6 How It Works 656
400.7 Remarks 656
400.8 Example 656
401 Use MSWC Components in G3Pix AxonASP 657
401.1 Overview 657
401.2 Supported ProgIDs 657
401.3 Prerequisites 657
401.4 How It Works 657
401.5 Code Example 657
402 MSWC Methods 658
402.1 Overview 658
402.2 Methods Reference 658
402.3 Remarks 662
403 MSWC Properties 662
403.1 Overview 662
403.2 Properties Reference 662
403.3 Remarks 664
404 AdRotator.GetAdvertisement Method 664
404.1 Overview 664

404.2 Syntax 664
404.3 Parameters and Arguments 664
404.4 Return Values 665
404.5 Remarks 665
404.6 Code Example 665
405 NextLink.GetListCount Method 665
405.1 Overview 665
405.2 Syntax 665
405.3 Parameters and Arguments 665
405.4 Return Values 665
405.5 Remarks 665
405.6 Code Example 666
406 NextLink.GetListIndex Method 666
406.1 Overview 666
406.2 Syntax 666
406.3 Parameters and Arguments 666
406.4 Return Values 666
406.5 Remarks 666
406.6 Code Example 666
407 NextLink.GetNextDescription Method 667
407.1 Overview 667
407.2 Syntax 667
407.3 Parameters and Arguments 667
407.4 Return Values 667
407.5 Remarks 667
407.6 Code Example 667
408 NextLink.GetNextURL Method 667
408.1 Overview 667
408.2 Syntax 668
408.3 Parameters and Arguments 668
408.4 Return Values 668
408.5 Remarks 668

408.6 Code Example 668
409 NextLink.GetPreviousDescription Method 668
409.1 Overview 668
409.2 Syntax 668
409.3 Parameters and Arguments 668
409.4 Return Values 669
409.5 Remarks 669
409.6 Code Example 669
410 NextLink.GetPreviousURL Method 669
410.1 Overview 669
410.2 Syntax 669
410.3 Parameters and Arguments 669
410.4 Return Values 669
410.5 Remarks 669
410.6 Code Example 670
411 NextLink.GetNthDescription Method 670
411.1 Overview 670
411.2 Syntax 670
411.3 Parameters and Arguments 670
411.4 Return Values 670
411.5 Remarks 670
411.6 Code Example 670
412 NextLink.GetNthURL Method 671
412.1 Overview 671
412.2 Syntax 671
412.3 Parameters and Arguments 671
412.4 Return Values 671
412.5 Remarks 671
412.6 Code Example 671
413 ContentRotator.ChooseContent Method 671
413.1 Overview 671
413.2 Syntax 672

413.3 Parameters and Arguments 672
413.4 Return Values 672
413.5 Remarks 672
413.6 Code Example 672
414 ContentRotator.GetAllContent Method 672
414.1 Overview 672
414.2 Syntax 672
414.3 Parameters and Arguments 672
414.4 Return Values 673
414.5 Remarks 673
414.6 Code Example 673
415 Counters.Get Method 673
415.1 Overview 673
415.2 Syntax 673
415.3 Parameters and Arguments 673
415.4 Return Values 673
415.5 Remarks 673
415.6 Code Example 674
416 Counters.Increment Method 674
416.1 Overview 674
416.2 Syntax 674
416.3 Parameters and Arguments 674
416.4 Return Values 674
416.5 Remarks 674
416.6 Code Example 674
417 Counters.Remove Method 675
417.1 Overview 675
417.2 Syntax 675
417.3 Parameters and Arguments 675
417.4 Return Values 675
417.5 Remarks 675
417.6 Code Example 675

418 Counters.Set Method 675
418.1 Overview 675
418.2 Syntax 676
418.3 Parameters and Arguments 676
418.4 Return Values 676
418.5 Remarks 676
418.6 Code Example 676
419 Tools.FileExists Method 676
419.1 Overview 676
419.2 Syntax 676
419.3 Parameters and Arguments 676
419.4 Return Values 677
419.5 Remarks 677
419.6 Code Example 677
420 Tools.Owner Method 677
420.1 Overview 677
420.2 Syntax 677
420.3 Parameters and Arguments 677
420.4 Return Values 677
420.5 Remarks 677
420.6 Code Example 678
421 Tools.PluginExists Method 678
421.1 Overview 678
421.2 Syntax 678
421.3 Parameters and Arguments 678
421.4 Return Values 678
421.5 Remarks 678
421.6 Code Example 678
422 Tools.ProcessForm Method 679
422.1 Overview 679
422.2 Syntax 679
422.3 Parameters and Arguments 679

422.4 Return Values 679
422.5 Remarks 679
422.6 Code Example 679
423 MyInfo.URL Method 679
423.1 Overview 679
423.2 Syntax 680
423.3 Parameters and Arguments 680
423.4 Return Values 680
423.5 Remarks 680
423.6 Code Example 680
424 MyInfo.URLWords Method 680
424.1 Overview 680
424.2 Syntax 680
424.3 Parameters and Arguments 680
424.4 Return Values 681
424.5 Remarks 681
424.6 Code Example 681
425 PageCounter.Hits Method 681
425.1 Overview 681
425.2 Syntax 681
425.3 Parameters and Arguments 681
425.4 Return Values 681
425.5 Remarks 681
425.6 Code Example 682
426 PageCounter.PageHit Method 682
426.1 Overview 682
426.2 Syntax 682
426.3 Parameters and Arguments 682
426.4 Return Values 682
426.5 Remarks 682
426.6 Code Example 682
427 PageCounter.Reset Method 683

427.1 Overview 683
427.2 Syntax 683
427.3 Parameters and Arguments 683
427.4 Return Values 683
427.5 Remarks 683
427.6 Code Example 683
428 PermissionChecker.HasAccess Method 683
428.1 Overview 683
428.2 Syntax 684
428.3 Parameters and Arguments 684
428.4 Return Values 684
428.5 Remarks 684
428.6 Code Example 684
429 AdRotator.Border Property 684
429.1 Overview 684
429.2 Syntax 684
429.3 Access 684
429.4 Return Values 685
429.5 Remarks 685
429.6 Code Example 685
430 AdRotator.Clickable Property 685
430.1 Overview 685
430.2 Syntax 685
430.3 Access 685
430.4 Return Values 685
430.5 Remarks 685
430.6 Code Example 685
431 AdRotator.TargetFrame Property 686
431.1 Overview 686
431.2 Syntax 686
431.3 Access 686
431.4 Return Values 686

431.5 Remarks 686
431.6 Code Example 686
432 BrowserType.Browser Property 686
432.1 Overview 686
432.2 Syntax 686
432.3 Access 686
432.4 Return Values 687
432.5 Remarks 687
432.6 Code Example 687
433 BrowserType.Version Property 687
433.1 Overview 687
433.2 Syntax 687
433.3 Access 687
433.4 Return Values 687
433.5 Remarks 687
433.6 Code Example 687
434 BrowserType.MajorVer Property 688
434.1 Overview 688
434.2 Syntax 688
434.3 Access 688
434.4 Return Values 688
434.5 Remarks 688
434.6 Code Example 688
435 BrowserType.MinorVer Property 688
435.1 Overview 688
435.2 Syntax 688
435.3 Access 688
435.4 Return Values 689
435.5 Remarks 689
435.6 Code Example 689
436 BrowserType.Frames Property 689
436.1 Overview 689

436.2 Syntax 689
436.3 Access 689
436.4 Return Values 689
436.5 Remarks 689
436.6 Code Example 689
437 BrowserType.Tables Property 690
437.1 Overview 690
437.2 Syntax 690
437.3 Access 690
437.4 Return Values 690
437.5 Remarks 690
437.6 Code Example 690
438 BrowserType.Cookies Property 690
438.1 Overview 690
438.2 Syntax 690
438.3 Access 690
438.4 Return Values 691
438.5 Remarks 691
438.6 Code Example 691
439 BrowserType.BackgroundSounds Property 691
439.1 Overview 691
439.2 Syntax 691
439.3 Access 691
439.4 Return Values 691
439.5 Remarks 691
439.6 Code Example 691
440 BrowserType.VBScript Property 692
440.1 Overview 692
440.2 Syntax 692
440.3 Access 692
440.4 Return Values 692
440.5 Remarks 692

440.6 Code Example 692
441 BrowserType.JavaScript Property 692
441.1 Overview 692
441.2 Syntax 692
441.3 Access 693
441.4 Return Values 693
441.5 Remarks 693
441.6 Code Example 693
442 BrowserType.JavaApplets Property 693
442.1 Overview 693
442.2 Syntax 693
442.3 Access 693
442.4 Return Values 693
442.5 Remarks 693
442.6 Code Example 693
443 BrowserType.ActiveXControls Property 694
443.1 Overview 694
443.2 Syntax 694
443.3 Access 694
443.4 Return Values 694
443.5 Remarks 694
443.6 Code Example 694
444 BrowserType.CDF Property 694
444.1 Overview 694
444.2 Syntax 694
444.3 Access 695
444.4 Return Values 695
444.5 Remarks 695
444.6 Code Example 695
445 MyInfo.AnyCustomProperty Property 695
445.1 Overview 695
445.2 Syntax 695

445.3 Access 695
445.4 Return Values 695
445.5 Remarks 695
445.6 Code Example 695
446 ADODB Library 696
446.1 Supported Objects 696
446.2 Supported ProgIDs 696
446.3 Prerequisites 696
446.4 How It Works 696
446.5 Code Example 697
447 ADODB Methods Reference 697
447.1 Connection Methods 697
447.2 Recordset Methods 698
447.3 Command Methods 700
447.4 Collection Methods 701
447.5 Code Example 702
448 ADODB Properties Reference 702
448.1 Connection Properties 702
448.2 Recordset Properties 703
448.3 Command Properties 706
448.4 Error Properties 706
448.5 Errors Collection Properties 707
448.6 Field Properties 707
448.7 Fields Collection Properties 708
448.8 Parameter Properties 708
448.9 Parameters Collection Properties 708
449 Connection.Open Method 709
449.1 Syntax 709
449.2 Parameters 709
449.3 Return Value 709
449.4 Remarks 709
449.5 Code Example 709

450 Connection.Close Method 710
450.1 Syntax 710
450.2 Parameters 710
450.3 Return Value 710
450.4 Remarks 710
450.5 Code Example 710
451 Connection.Execute Method 710
451.1 Syntax 710
451.2 Parameters 711
451.3 Return Value 711
451.4 Remarks 711
451.5 Code Example 711
452 Connection.BeginTrans Method 711
452.1 Syntax 711
452.2 Parameters 712
452.3 Return Value 712
452.4 Remarks 712
452.5 Code Example 712
453 Connection.CommitTrans Method 712
453.1 Syntax 712
453.2 Parameters 712
453.3 Return Value 713
453.4 Remarks 713
453.5 Code Example 713
454 Connection.RollbackTrans Method 713
454.1 Syntax 713
454.2 Parameters 713
454.3 Return Value 713
454.4 Remarks 713
454.5 Code Example 714
455 Connection.OpenSchema Method 714
455.1 Syntax 714

455.2 Parameters 714
455.3 Return Value 715
455.4 Remarks 715
455.5 Code Example 715
456 Connection.Cancel Method 715
456.1 Syntax 715
456.2 Parameters 715
456.3 Return Value 715
456.4 Remarks 716
456.5 Code Example 716
457 Recordset.Open Method 716
457.1 Syntax 716
457.2 Parameters 716
457.3 Return Value 717
457.4 Remarks 717
457.5 Code Example 717
458 Recordset.Close Method 717
458.1 Syntax 717
458.2 Parameters 717
458.3 Return Value 718
458.4 Remarks 718
458.5 Code Example 718
459 Recordset.MoveNext Method 718
459.1 Syntax 718
459.2 Parameters 718
459.3 Return Value 718
459.4 Remarks 718
459.5 Code Example 719
460 Recordset.MovePrevious Method 719
460.1 Syntax 719
460.2 Parameters 719
460.3 Return Value 719

460.4 Remarks 719
460.5 Code Example 720
461 Recordset.MoveFirst Method 720
461.1 Syntax 720
461.2 Parameters 720
461.3 Return Value 720
461.4 Remarks 720
461.5 Code Example 721
462 Recordset.MoveLast Method 721
462.1 Syntax 721
462.2 Parameters 721
462.3 Return Value 721
462.4 Remarks 721
462.5 Code Example 721
463 Recordset.Move Method 722
463.1 Syntax 722
463.2 Parameters 722
463.3 Return Value 722
463.4 Remarks 722
463.5 Code Example 722
464 Recordset.Cancel Method 723
464.1 Syntax 723
464.2 Parameters 723
464.3 Return Value 723
464.4 Remarks 723
464.5 Code Example 723
465 Recordset.CancelBatch Method 724
465.1 Syntax 724
465.2 Parameters 724
465.3 Return Value 724
465.4 Remarks 724
465.5 Code Example 724

466 Recordset.CancelUpdate Method 725
466.1 Syntax 725
466.2 Parameters 725
466.3 Return Value 725
466.4 Remarks 725
466.5 Code Example 725
467 Recordset.Find Method 726
467.1 Syntax 726
467.2 Parameters 726
467.3 Return Value 726
467.4 Remarks 726
467.5 Code Example 727
468 Recordset.NextRecordset Method 727
468.1 Syntax 727
468.2 Parameters 727
468.3 Return Value 727
468.4 Remarks 727
468.5 Code Example 728
469 Recordset.Clone Method 728
469.1 Syntax 728
469.2 Parameters 728
469.3 Return Value 728
469.4 Remarks 728
469.5 Code Example 729
470 Recordset.CompareBookmarks Method 729
470.1 Syntax 729
470.2 Parameters 729
470.3 Return Value 730
470.4 Remarks 730
470.5 Code Example 730
471 Recordset.AddNew Method 730
471.1 Syntax 730

471.2 Parameters 731
471.3 Return Value 731
471.4 Remarks 731
471.5 Code Example 731
472 Recordset.Update Method 731
472.1 Syntax 731
472.2 Parameters 731
472.3 Return Value 732
472.4 Remarks 732
472.5 Code Example 732
473 Recordset.Delete Method 732
473.1 Syntax 732
473.2 Parameters 732
473.3 Return Value 732
473.4 Remarks 733
473.5 Code Example 733
474 Recordset.GetRows Method 733
474.1 Syntax 733
474.2 Parameters 733
474.3 Return Value 733
474.4 Remarks 733
474.5 Code Example 734
475 Recordset.GetString Method 734
475.1 Syntax 734
475.2 Parameters 734
475.3 Return Value 734
475.4 Remarks 734
475.5 Code Example 735
476 Recordset.Requery Method 735
476.1 Syntax 735
476.2 Parameters 735
476.3 Return Value 735

476.4 Remarks 735
476.5 Code Example 736
477 Recordset.Resync Method 736
477.1 Syntax 736
477.2 Parameters 736
477.3 Return Value 736
477.4 Remarks 736
477.5 Code Example 737
478 Recordset.Save Method 737
478.1 Syntax 737
478.2 Parameters 737
478.3 Return Value 737
478.4 Remarks 737
478.5 Code Example 738
479 Recordset.Seek Method 738
479.1 Syntax 738
479.2 Parameters 738
479.3 Return Value 738
479.4 Remarks 738
479.5 Code Example 739
480 Recordset.Supports Method 739
480.1 Syntax 739
480.2 Parameters 739
480.3 Return Value 739
480.4 Remarks 739
480.5 Code Example 739
481 Recordset.UpdateBatch Method 740
481.1 Syntax 740
481.2 Parameters 740
481.3 Return Value 740
481.4 Remarks 740
481.5 Code Example 740

482 Recordset.Fields Method 741
482.1 Syntax 741
482.2 Parameters 741
482.3 Return Value 741
482.4 Remarks 741
482.5 Code Example 741
483 Recordset.Fields.Append Method 742
483.1 Syntax 742
483.2 Parameters 742
483.3 Return Value 742
483.4 Remarks 742
483.5 Code Example 742
484 Recordset.Fields.Item Method 743
484.1 Syntax 743
484.2 Parameters 743
484.3 Return Value 743
484.4 Remarks 743
484.5 Code Example 743
485 Command.Execute Method 744
485.1 Syntax 744
485.2 Parameters 744
485.3 Return Value 744
485.4 Remarks 744
485.5 Code Example 744
486 Command.CreateParameter Method 745
486.1 Syntax 745
486.2 Parameters 745
486.3 Return Value 745
486.4 Remarks 745
486.5 Code Example 746
487 Command.Cancel Method 746
487.1 Syntax 746

487.2 Parameters 746
487.3 Return Value 746
487.4 Remarks 746
487.5 Code Example 747
488 Command.Parameters.Refresh Method 747
488.1 Syntax 747
488.2 Parameters 747
488.3 Return Value 747
488.4 Remarks 748
488.5 Code Example 748
489 Errors.Count Method 748
489.1 Syntax 748
489.2 Parameters 748
489.3 Return Value 749
489.4 Remarks 749
489.5 Code Example 749
490 Errors.Item Method 749
490.1 Syntax 749
490.2 Parameters 749
490.3 Return Value 750
490.4 Remarks 750
490.5 Code Example 750
491 Errors.Clear Method 750
491.1 Syntax 750
491.2 Parameters 750
491.3 Return Value 751
491.4 Remarks 751
491.5 Code Example 751
492 Fields.Append Method 751
492.1 Syntax 751
492.2 Parameters 751
492.3 Return Value 752

492.4 Remarks 752
492.5 Code Example 752
493 Fields.Item Method 752
493.1 Syntax 752
493.2 Parameters 752
493.3 Return Value 753
493.4 Remarks 753
493.5 Code Example 753
494 Fields.Count Method 753
494.1 Syntax 753
494.2 Parameters 753
494.3 Return Value 754
494.4 Remarks 754
494.5 Code Example 754
495 Field.AppendChunk Method 754
495.1 Syntax 754
495.2 Parameters 754
495.3 Return Value 755
495.4 Remarks 755
495.5 Code Example 755
496 Field.GetChunk Method 755
496.1 Syntax 755
496.2 Parameters 755
496.3 Return Value 756
496.4 Remarks 756
496.5 Code Example 756
497 Parameters.Append Method 756
497.1 Syntax 756
497.2 Parameters 757
497.3 Return Value 757
497.4 Remarks 757
497.5 Code Example 757

498 Parameters.Count Method 757
498.1 Syntax 758
498.2 Parameters 758
498.3 Return Value 758
498.4 Remarks 758
498.5 Code Example 758
499 Connection.ConnectionString Property 758
499.1 Syntax 759
499.2 Return Value 759
499.3 Remarks 759
499.4 Code Example 759
500 Connection.State Property 759
500.1 Syntax 759
500.2 Return Value 759
500.3 Remarks 759
500.4 Code Example 760
501 Connection.Mode Property 760
501.1 Syntax 760
501.2 Return Value 760
501.3 Remarks 760
501.4 Code Example 760
502 Connection.Provider Property 761
502.1 Syntax 761
502.2 Return Value 761
502.3 Remarks 761
502.4 Code Example 761
503 Connection.Version Property 761
503.1 Syntax 761
503.2 Return Value 762
503.3 Remarks 762
503.4 Code Example 762
504 Connection.CommandTimeout Property 762

504.1 Syntax 762
504.2 Return Value 762
504.3 Remarks 762
504.4 Code Example 762
505 Connection.ConnectionTimeout Property 763
505.1 Syntax 763
505.2 Return Value 763
505.3 Remarks 763
505.4 Code Example 763
506 Connection.CursorLocation Property 764
506.1 Syntax 764
506.2 Return Value 764
506.3 Remarks 764
506.4 Code Example 764
507 Connection.DefaultDatabase Property 764
507.1 Syntax 764
507.2 Return Value 764
507.3 Remarks 765
507.4 Code Example 765
508 Connection.IsolationLevel Property 765
508.1 Syntax 765
508.2 Return Value 765
508.3 Remarks 765
508.4 Code Example 765
509 Connection.Errors Property 766
509.1 Syntax 766
509.2 Return Value 766
509.3 Remarks 766
509.4 Code Example 766
510 Recordset.EOF Property 767
510.1 Syntax 767
510.2 Return Value 767

510.3 Remarks 767
510.4 Code Example 767
511 Recordset.BOF Property 767
511.1 Syntax 767
511.2 Return Value 768
511.3 Remarks 768
511.4 Code Example 768
512 Recordset.State Property 768
512.1 Syntax 768
512.2 Return Value 768
512.3 Remarks 768
512.4 Code Example 769
513 Recordset.RecordCount Property 769
513.1 Syntax 769
513.2 Return Value 769
513.3 Remarks 769
513.4 Code Example 769
514 Recordset.Fields Property 770
514.1 Syntax 770
514.2 Return Value 770
514.3 Remarks 770
514.4 Code Example 770
515 Recordset.AbsolutePage Property 771
515.1 Syntax 771
515.2 Return Value 771
515.3 Remarks 771
515.4 Code Example 771
516 Recordset.PageCount Property 771
516.1 Syntax 771
516.2 Return Value 772
516.3 Remarks 772
516.4 Code Example 772

517 Recordset.PageSize Property 772
517.1 Syntax 772
517.2 Return Value 772
517.3 Remarks 772
517.4 Code Example 773
518 Recordset.AbsolutePosition Property 773
518.1 Syntax 773
518.2 Return Value 773
518.3 Remarks 773
518.4 Code Example 773
519 Recordset.Bookmark Property 774
519.1 Syntax 774
519.2 Return Value 774
519.3 Remarks 774
519.4 Code Example 774
520 Recordset.ActiveCommand Property 774
520.1 Syntax 775
520.2 Return Value 775
520.3 Remarks 775
520.4 Code Example 775
521 Recordset.CacheSize Property 775
521.1 Syntax 775
521.2 Return Value 776
521.3 Remarks 776
521.4 Code Example 776
522 Recordset.DataMember Property 776
522.1 Syntax 776
522.2 Return Value 776
522.3 Remarks 776
522.4 Code Example 777
523 Recordset.EditMode Property 777
523.1 Syntax 777

523.2 Return Value 777
523.3 Remarks 777
523.4 Code Example 777
524 Recordset.Filter Property 778
524.1 Syntax 778
524.2 Return Value 778
524.3 Remarks 778
524.4 Code Example 778
525 Recordset.Index Property 778
525.1 Syntax 779
525.2 Return Value 779
525.3 Remarks 779
525.4 Code Example 779
526 Recordset.MarshalOptions Property 779
526.1 Syntax 779
526.2 Return Value 779
526.3 Remarks 780
526.4 Code Example 780
527 Recordset.MaxRecords Property 780
527.1 Syntax 780
527.2 Return Value 780
527.3 Remarks 780
527.4 Code Example 780
528 Recordset.Sort Property 781
528.1 Syntax 781
528.2 Return Value 781
528.3 Remarks 781
528.4 Code Example 781
529 Recordset.Source Property 781
529.1 Syntax 781
529.2 Return Value 782
529.3 Remarks 782

529.4 Code Example 782
530 Recordset.Status Property 782
530.1 Syntax 782
530.2 Return Value 782
530.3 Remarks 782
530.4 Code Example 782
531 Recordset.ActiveConnection Property 783
531.1 Syntax 783
531.2 Return Value 783
531.3 Remarks 783
531.4 Code Example 783
532 Recordset.CursorType Property 784
532.1 Syntax 784
532.2 Return Value 784
532.3 Remarks 784
532.4 Code Example 784
533 Recordset.LockType Property 784
533.1 Syntax 784
533.2 Return Value 785
533.3 Remarks 785
533.4 Code Example 785
534 Recordset.CursorLocation Property 785
534.1 Syntax 785
534.2 Return Value 785
534.3 Remarks 785
534.4 Code Example 785
535 Command.ActiveConnection Property 786
535.1 Syntax 786
535.2 Return Value 786
535.3 Remarks 786
535.4 Code Example 786
536 Command.CommandText Property 787

536.1 Syntax 787
536.2 Return Value 787
536.3 Remarks 787
536.4 Code Example 787
537 Command.CommandType Property 787
537.1 Syntax 788
537.2 Return Value 788
537.3 Remarks 788
537.4 Code Example 788
538 Command.CommandTimeout Property 788
538.1 Syntax 788
538.2 Return Value 789
538.3 Remarks 789
538.4 Code Example 789
539 Command.Prepared Property 789
539.1 Syntax 789
539.2 Return Value 789
539.3 Remarks 790
539.4 Code Example 790
540 Command.Parameters Property 790
540.1 Syntax 790
540.2 Return Value 790
540.3 Remarks 790
540.4 Code Example 791
541 Error.Number Property 791
541.1 Syntax 791
541.2 Return Value 791
541.3 Remarks 791
541.4 Code Example 791
542 Error.Description Property 792
542.1 Syntax 792
542.2 Return Value 792

542.3 Remarks 792
542.4 Code Example 792
543 Error.Source Property 793
543.1 Syntax 793
543.2 Return Value 793
543.3 Remarks 793
543.4 Code Example 793
544 Error.SQLState Property 794
544.1 Syntax 794
544.2 Return Value 794
544.3 Remarks 794
544.4 Code Example 794
545 Field.Value Property 795
545.1 Syntax 795
545.2 Return Value 795
545.3 Remarks 795
545.4 Code Example 795
546 Field.Name Property 796
546.1 Syntax 796
546.2 Return Value 796
546.3 Remarks 796
546.4 Code Example 796
547 Field.Type Property 796
547.1 Syntax 796
547.2 Return Value 797
547.3 Remarks 797
547.4 Code Example 797
548 Field.DefinedSize Property 797
548.1 Syntax 797
548.2 Return Value 797
548.3 Remarks 797
548.4 Code Example 798

549 Field.Attributes Property 798
549.1 Syntax 798
549.2 Return Value 798
549.3 Remarks 798
549.4 Code Example 798
550 Field.NumericScale Property 799
550.1 Syntax 799
550.2 Return Value 799
550.3 Remarks 799
550.4 Code Example 799
551 Field.ActualSize Property 800
551.1 Syntax 800
551.2 Return Value 800
551.3 Remarks 800
551.4 Code Example 800
552 Field.DataFormat Property 801
552.1 Syntax 801
552.2 Return Value 801
552.3 Remarks 801
552.4 Code Example 801
553 Field.OriginalValue Property 801
553.1 Syntax 802
553.2 Return Value 802
553.3 Remarks 802
553.4 Code Example 802
554 Field.Precision Property 802
554.1 Syntax 802
554.2 Return Value 803
554.3 Remarks 803
554.4 Code Example 803
555 Field.Status Property 803
555.1 Syntax 803

555.2 Return Value 803
555.3 Remarks 803
555.4 Code Example 804
556 Field.UnderlyingValue Property 804
556.1 Syntax 804
556.2 Return Value 804
556.3 Remarks 804
556.4 Code Example 804
557 Errors.Count Property 805
557.1 Syntax 805
557.2 Return Value 805
557.3 Remarks 805
557.4 Code Example 805
558 Errors.Item Property 806
558.1 Syntax 806
558.2 Return Value 806
558.3 Remarks 806
558.4 Code Example 806
559 Fields.Count Property 806
559.1 Syntax 807
559.2 Return Value 807
559.3 Remarks 807
559.4 Code Example 807
560 Fields.Item Property 807
560.1 Syntax 807
560.2 Return Value 807
560.3 Remarks 807
560.4 Code Example 808
561 Parameters.Count Property 808
561.1 Syntax 808
561.2 Return Value 808
561.3 Remarks 808

561.4 Code Example 808
562 Parameters.Item Property 809
562.1 Syntax 809
562.2 Return Value 809
562.3 Remarks 809
562.4 Code Example 809
563 Use ADODB.Stream in G3Pix AxonASP 810
563.1 Overview 810
563.2 Prerequisites 810
563.3 Syntax 810
563.4 Return Value 810
563.5 How It Works 810
563.6 Remarks 810
563.7 Example 811
564 ADODB.Stream Methods Reference 811
564.1 Overview 811
564.2 Methods 811
564.3 How It Works 812
564.4 Remarks 812
564.5 Example 812
565 ADODB.Stream Properties Reference 813
565.1 Overview 813
565.2 Properties 813
565.3 How It Works 814
565.4 Remarks 814
566 Open Method 814
566.1 Overview 814
566.2 Prerequisites 814
566.3 Syntax 814
566.4 Parameters 814
566.5 Return Value 814
566.6 How It Works 814

566.7 Remarks 815
566.8 Code Example 815
566.9 API Reference 815
567 Close Method 815
567.1 Overview 815
567.2 Prerequisites 815
567.3 Syntax 815
567.4 Parameters 815
567.5 Return Value 816
567.6 How It Works 816
567.7 Remarks 816
567.8 Code Example 816
567.9 API Reference 816
568 Read Method 816
568.1 Overview 816
568.2 Prerequisites 816
568.3 Syntax 817
568.4 Parameters 817
568.5 Return Value 817
568.6 How It Works 817
568.7 Remarks 817
568.8 Code Example 817
568.9 API Reference 818
569 ReadText Method 818
569.1 Overview 818
569.2 Prerequisites 818
569.3 Syntax 818
569.4 Parameters 818
569.5 Return Value 818
569.6 How It Works 818
569.7 Remarks 818
569.8 Code Example 819

569.9 API Reference 819
570 Write Method 819
570.1 Overview 819
570.2 Prerequisites 819
570.3 Syntax 819
570.4 Parameters 819
570.5 Return Value 820
570.6 How It Works 820
570.7 Remarks 820
570.8 Code Example 820
570.9 API Reference 820
571 WriteText Method 820
571.1 Overview 820
571.2 Prerequisites 821
571.3 Syntax 821
571.4 Parameters 821
571.5 Return Value 821
571.6 How It Works 821
571.7 Remarks 821
571.8 Code Example 821
571.9 API Reference 822
572 LoadFromFile Method 822
572.1 Overview 822
572.2 Prerequisites 822
572.3 Syntax 822
572.4 Parameters 822
572.5 Return Value 822
572.6 How It Works 822
572.7 Remarks 823
572.8 Code Example 823
572.9 API Reference 823
573 SaveToFile Method 823

573.1 Overview 823
573.2 Prerequisites 823
573.3 Syntax 823
573.4 Parameters 823
573.5 Return Value 824
573.6 How It Works 824
573.7 Remarks 824
573.8 Code Example 824
573.9 API Reference 824
574 CopyTo Method 825
574.1 Overview 825
574.2 Prerequisites 825
574.3 Syntax 825
574.4 Parameters 825
574.5 Return Value 825
574.6 How It Works 825
574.7 Remarks 826
574.8 Code Example 826
574.9 API Reference 826
575 Flush Method 826
575.1 Overview 826
575.2 Prerequisites 827
575.3 Syntax 827
575.4 Parameters 827
575.5 Return Value 827
575.6 How It Works 827
575.7 Remarks 827
575.8 Code Example 827
575.9 API Reference 827
576 SetEOS Method 828
576.1 Overview 828
576.2 Prerequisites 828

576.3 Syntax 828
576.4 Parameters 828
576.5 Return Value 828
576.6 How It Works 828
576.7 Remarks 828
576.8 Code Example 828
576.9 API Reference 829
577 SkipLine Method 829
577.1 Overview 829
577.2 Prerequisites 829
577.3 Syntax 829
577.4 Parameters 829
577.5 Return Value 829
577.6 How It Works 829
577.7 Remarks 829
577.8 Code Example 829
577.9 API Reference 830
578 Type Property 830
578.1 Overview 830
578.2 Syntax 830
578.3 Return Value 830
578.4 Remarks 830
578.5 Code Example 830
578.6 API Reference 831
579 Mode Property 831
579.1 Overview 831
579.2 Syntax 831
579.3 Return Value 831
579.4 Remarks 831
579.5 Code Example 831
579.6 API Reference 832
580 State Property 832

580.1 Overview 832
580.2 Syntax 832
580.3 Return Value 832
580.4 Remarks 832
580.5 Code Example 832
580.6 API Reference 833
581 Position Property 833
581.1 Overview 833
581.2 Syntax 833
581.3 Return Value 833
581.4 Remarks 833
581.5 Code Example 833
581.6 API Reference 834
582 Size Property 834
582.1 Overview 834
582.2 Syntax 834
582.3 Return Value 834
582.4 Remarks 834
582.5 Code Example 834
582.6 API Reference 834
583 Charset Property 835
583.1 Overview 835
583.2 Syntax 835
583.3 Return Value 835
583.4 Remarks 835
583.5 Code Example 835
583.6 API Reference 835
584 LineSeparator Property 836
584.1 Overview 836
584.2 Syntax 836
584.3 Return Value 836
584.4 Remarks 836

584.5 Code Example 836
584.6 API Reference 836
585 EOS Property 837
585.1 Overview 837
585.2 Syntax 837
585.3 Return Value 837
585.4 Remarks 837
585.5 Code Example 837
585.6 API Reference 837
586 Use Scripting.Dictionary in G3Pix AxonASP 838
586.1 Overview 838
586.2 Prerequisites 838
586.3 Syntax 838
586.4 Parameters and Arguments 838
586.5 Return Value 838
586.6 How It Works 838
586.7 Remarks 838
586.8 Code Example 838
587 Scripting.Dictionary Methods 839
587.1 Overview 839
587.2 Methods Reference 839
587.3 Remarks 840
588 Scripting.Dictionary Properties 840
588.1 Overview 840
588.2 Properties Reference 840
588.3 Remarks 840
589 Add a Dictionary Entry 841
589.1 Overview 841
589.2 Syntax 841
589.3 Parameters 841
589.4 Return Value 841
589.5 How It Works 841

589.6 Remarks 841
589.7 Example 841
590 Count Dictionary Entries 841
590.1 Overview 841
590.2 Syntax 842
590.3 Parameters 842
590.4 Return Value 842
590.5 How It Works 842
590.6 Remarks 842
590.7 Example 842
591 Check Whether a Key Exists 842
591.1 Overview 842
591.2 Syntax 842
591.3 Parameters 842
591.4 Return Value 843
591.5 How It Works 843
591.6 Remarks 843
591.7 Example 843
592 Get or Set a Dictionary Item 843
592.1 Overview 843
592.2 Syntax 843
592.3 Parameters 843
592.4 Return Value 843
592.5 How It Works 844
592.6 Remarks 844
592.7 Example 844
593 Enumerate Dictionary Values 844
593.1 Overview 844
593.2 Syntax 844
593.3 Parameters 844
593.4 Return Value 844
593.5 How It Works 845

593.6 Remarks 845
593.7 Example 845
594 Rename a Dictionary Key 845
594.1 Overview 845
594.2 Syntax 845
594.3 Parameters 845
594.4 Return Value 845
594.5 How It Works 845
594.6 Remarks 846
594.7 Example 846
595 Enumerate Dictionary Keys 846
595.1 Overview 846
595.2 Syntax 846
595.3 Parameters 846
595.4 Return Value 846
595.5 How It Works 846
595.6 Remarks 846
595.7 Example 847
596 Remove a Dictionary Entry 847
596.1 Overview 847
596.2 Syntax 847
596.3 Parameters 847
596.4 Return Value 847
596.5 How It Works 847
596.6 Remarks 847
596.7 Example 847
597 Clear All Dictionary Entries 848
597.1 Overview 848
597.2 Syntax 848
597.3 Parameters 848
597.4 Return Value 848
597.5 How It Works 848

597.6 Remarks 848
597.7 Example 848
598 Set Dictionary CompareMode 849
598.1 Overview 849
598.2 Syntax 849
598.3 Parameters 849
598.4 Return Value 849
598.5 How It Works 849
598.6 Remarks 849
598.7 Example 849
599 Get Dictionary Entry Count 850
599.1 Overview 850
599.2 Syntax 850
599.3 Parameters 850
599.4 Return Value 850
599.5 How It Works 850
599.6 Remarks 850
599.7 Example 850
600 Use Scripting.FileSystemObject in AxonASP 851
600.1 Overview 851
600.2 Prerequisites 851
600.3 Syntax 851
600.4 Return Value 851
600.5 How It Works 851
600.6 Remarks 852
600.7 Code Example 852
601 Methods 853
601.1 Overview 853
601.2 FileSystemObject Methods 853
601.3 Remarks 856
602 Properties 856
602.1 Overview 856

602.2 FileSystemObject Properties 856
602.3 File Properties 856
602.4 Folder Properties 858
602.5 TextStream Properties 859
602.6 Drive Properties 860
602.7 Remarks 861
603 BuildPath Method 861
603.1 Overview 861
603.2 Syntax 861
603.3 Parameters 861
603.4 Return Value 862
603.5 How It Works 862
603.6 Remarks 862
603.7 Code Example 862
604 CopyFile Method 862
604.1 Overview 862
604.2 Syntax 862
604.3 Parameters 862
604.4 Return Value 863
604.5 How It Works 863
604.6 Remarks 863
604.7 Code Example 863
605 CopyFolder Method 864
605.1 Overview 864
605.2 Syntax 864
605.3 Parameters 864
605.4 Return Value 864
605.5 How It Works 864
605.6 Remarks 864
605.7 Code Example 865
606 CreateFolder Method 865
606.1 Overview 865

606.2 Syntax 865
606.3 Parameters 865
606.4 Return Value 865
606.5 How It Works 865
606.6 Remarks 865
606.7 Code Example 866
607 CreateTextFile Method 866
607.1 Overview 866
607.2 Syntax 866
607.3 Parameters 866
607.4 Return Value 867
607.5 How It Works 867
607.6 Remarks 867
607.7 Code Example 867
608 DeleteFile Method 867
608.1 Overview 867
608.2 Syntax 867
608.3 Parameters 868
608.4 Return Value 868
608.5 How It Works 868
608.6 Error Conditions 868
608.7 Remarks 868
608.8 Code Example 868
609 DeleteFolder Method 869
609.1 Overview 869
609.2 Syntax 869
609.3 Parameters 869
609.4 Return Value 869
609.5 How It Works 869
609.6 Error Conditions 869
609.7 Remarks 869
609.8 Code Example 869

610 FileExists Method 870
610.1 Overview 870
610.2 Syntax 870
610.3 Parameters 870
610.4 Return Value 870
610.5 How It Works 870
610.6 Remarks 870
610.7 Code Example 871
611 FolderExists Method 871
611.1 Overview 871
611.2 Syntax 871
611.3 Parameters 871
611.4 Return Value 871
611.5 How It Works 871
611.6 Remarks 871
611.7 Code Example 872
612 GetFile Method 872
612.1 Overview 872
612.2 Syntax 872
612.3 Parameters 872
612.4 Return Value 872
612.5 Error Conditions 872
612.6 How It Works 873
612.7 Remarks 873
612.8 Code Example 873
613 GetFolder Method 873
613.1 Overview 873
613.2 Syntax 873
613.3 Parameters 873
613.4 Return Value 873
613.5 Error Conditions 874
613.6 How It Works 874

613.7 Remarks 874
613.8 Code Example 874
614 GetTempName Method 874
614.1 Overview 874
614.2 Syntax 874
614.3 Parameters 874
614.4 Return Value 875
614.5 How It Works 875
614.6 Remarks 875
614.7 Code Example 875
615 MoveFile Method 875
615.1 Overview 875
615.2 Syntax 876
615.3 Parameters 876
615.4 Return Value 876
615.5 How It Works 876
615.6 Remarks 876
615.7 Code Example 876
616 MoveFolder Method 877
616.1 Overview 877
616.2 Syntax 877
616.3 Parameters 877
616.4 Return Value 877
616.5 How It Works 877
616.6 Remarks 877
616.7 Code Example 877
617 OpenTextFile Method 878
617.1 Overview 878
617.2 Syntax 878
617.3 Parameters 878
617.4 Return Value 878
617.5 How It Works 879

617.6 Remarks 879
617.7 Code Example 879
618 Use VBScript.RegExp in AxonASP 880
618.1 Overview 880
618.2 Prerequisites 880
618.3 Syntax 880
618.4 Return Value 880
618.5 How It Works 880
618.6 Remarks 881
618.7 Code Example 881
619 Methods 881
619.1 Overview 881
619.2 RegExp Methods 881
619.3 MatchesCollection Methods 882
619.4 Match Methods 882
619.5 SubMatches Methods 883
619.6 Remarks 883
620 Properties 883
620.1 Overview 883
620.2 RegExp Properties 883
620.3 Match Properties 885
620.4 MatchesCollection Properties 886
620.5 SubMatches Properties 886
620.6 SubMatchValue Properties 886
621 Execute Method 886
621.1 Overview 886
621.2 Syntax 886
621.3 Parameters 887
621.4 Return Value 887
621.5 How It Works 887
621.6 Remarks 887
621.7 Code Example 888

622 Replace Method 888
622.1 Overview 888
622.2 Syntax 888
622.3 Parameters 888
622.4 Return Value 888
622.5 How It Works 889
622.6 Remarks 889
622.7 Code Example 889
623 Test Method 890
623.1 Overview 890
623.2 Syntax 890
623.3 Parameters 890
623.4 Return Value 890
623.5 How It Works 890
623.6 Remarks 890
623.7 Code Example 890
624 Global Property 891
624.1 Overview 891
624.2 Syntax 891
624.3 Return Value 891
624.4 How It Works 891
624.5 Remarks 891
624.6 Code Example 892
625 IgnoreCase Property 892
625.1 Overview 892
625.2 Syntax 892
625.3 Return Value 892
625.4 How It Works 892
625.5 Remarks 892
625.6 Code Example 893
626 MultiLine Property 893
626.1 Overview 893

626.2 Syntax 893
626.3 Return Value 893
626.4 How It Works 893
626.5 Remarks 893
626.6 Code Example 894
627 Pattern Property 894
627.1 Overview 894
627.2 Syntax 894
627.3 Return Value 894
627.4 How It Works 894
627.5 Remarks 895
627.6 Code Example 895
628 MSXML2 Library 895
628.1 Supported ProgIDs 895
628.2 Prerequisites 896
628.3 Object Model 896
628.4 Remarks 896
628.5 Code Examples 896
629 MSXML2 Methods 897
629.1 ServerXMLHTTP Methods 897
629.2 DOMDocument Methods 897
629.3 XMLNodeList Methods 898
629.4 XMLElement Methods 899
629.5 Method List by Object 899
629.6 Remarks 900
630 MSXML2 Properties 900
630.1 ServerXMLHTTP Properties 900
630.2 DOMDocument Properties 901
630.3 XMLNodeList Properties 902
630.4 ParseError Properties 903
630.5 XMLElement Properties 903
631 ServerXMLHTTP.Open Method 904

631.1 Syntax 905
631.2 Parameters 905
631.3 Return Value 905
631.4 Remarks 905
631.5 Code Example 906
632 ServerXMLHTTP.SetRequestHeader Method 906
632.1 Syntax 906
632.2 Parameters 906
632.3 Return Value 906
632.4 Remarks 906
632.5 Code Example 906
633 ServerXMLHTTP.Send Method 907
633.1 Syntax 907
633.2 Parameters 907
633.3 Return Value 907
633.4 Remarks 907
633.5 Code Example 908
634 ServerXMLHTTP.Abort Method 908
634.1 Syntax 908
634.2 Parameters 908
634.3 Return Value 908
634.4 Remarks 908
634.5 Code Example 908
635 ServerXMLHTTP.GetResponseHeader Method 909
635.1 Syntax 909
635.2 Parameters 909
635.3 Return Value 909
635.4 Remarks 909
635.5 Code Example 909
636 ServerXMLHTTP.GetAllResponseHeaders Method 909
636.1 Syntax 909
636.2 Parameters 910

636.3 Return Value 910
636.4 Remarks 910
636.5 Code Example 910
637 DOMDocument.GetProperty Method 910
637.1 Syntax 910
637.2 Parameters 910
637.3 Return Value 911
637.4 Remarks 911
637.5 Code Example 911
638 DOMDocument.SetProperty Method 911
638.1 Syntax 911
638.2 Parameters 911
638.3 Return Value 911
638.4 Remarks 911
638.5 Code Example 912
639 DOMDocument.LoadXML Method 912
639.1 Syntax 912
639.2 Parameters 912
639.3 Return Value 912
639.4 Remarks 912
639.5 Code Example 912
640 DOMDocument.Load Method 913
640.1 Syntax 913
640.2 Parameters 913
640.3 Return Value 913
640.4 Remarks 913
640.5 Code Example 913
641 DOMDocument.Save Method 914
641.1 Syntax 914
641.2 Parameters 914
641.3 Return Value 914
641.4 Remarks 914

641.5 Code Example 914
642 DOMDocument.GetElementsByTagName Method 914
642.1 Syntax 915
642.2 Parameters 915
642.3 Return Value 915
642.4 Remarks 915
642.5 Code Example 915
643 DOMDocument.CreateElement Method 915
643.1 Syntax 915
643.2 Parameters 915
643.3 Return Value 916
643.4 Remarks 916
643.5 Code Example 916
644 DOMDocument.CreateTextNode Method 916
644.1 Syntax 916
644.2 Parameters 916
644.3 Return Value 916
644.4 Remarks 916
644.5 Code Example 916
645 DOMDocument.CreateAttribute Method 917
645.1 Syntax 917
645.2 Parameters 917
645.3 Return Value 917
645.4 Remarks 917
645.5 Code Example 917
646 DOMDocument.AppendChild Method 917
646.1 Syntax 917
646.2 Parameters 918
646.3 Return Value 918
646.4 Remarks 918
646.5 Code Example 918
647 DOMDocument.SelectSingleNode Method 918

647.1 Syntax 918
647.2 Parameters 918
647.3 Return Value 919
647.4 Remarks 919
647.5 Code Example 919
648 DOMDocument.SelectNodes Method 919
648.1 Syntax 919
648.2 Parameters 919
648.3 Return Value 919
648.4 Remarks 920
648.5 Code Example 920
649 XMLNodeList.Item Method 920
649.1 Syntax 920
649.2 Parameters 920
649.3 Return Value 920
649.4 Remarks 920
649.5 Code Example 921
650 XMLNodeList.NextNode Method 921
650.1 Syntax 921
650.2 Parameters 921
650.3 Return Value 921
650.4 Remarks 921
650.5 Code Example 921
651 XMLElement.AppendChild Method 922
651.1 Syntax 922
651.2 Parameters 922
651.3 Return Value 922
651.4 Remarks 922
651.5 Code Example 922
652 XMLElement.GetElementsByTagName Method 922
652.1 Syntax 922
652.2 Parameters 922

652.3 Return Value 923
652.4 Remarks 923
652.5 Code Example 923
653 XMLElement.Item Method 923
653.1 Syntax 923
653.2 Parameters 923
653.3 Return Value 924
653.4 Remarks 924
653.5 Code Example 924
654 XMLElement.SetAttribute Method 924
654.1 Syntax 924
654.2 Parameters 924
654.3 Return Value 924
654.4 Remarks 924
654.5 Code Example 924
655 XMLElement.GetAttribute Method 925
655.1 Syntax 925
655.2 Parameters 925
655.3 Return Value 925
655.4 Remarks 925
655.5 Code Example 925
656 XMLElement.RemoveAttribute Method 925
656.1 Syntax 925
656.2 Parameters 926
656.3 Return Value 926
656.4 Remarks 926
656.5 Code Example 926
657 XMLElement.SelectSingleNode Method 926
657.1 Syntax 926
657.2 Parameters 926
657.3 Return Value 926
657.4 Remarks 927

657.5 Code Example 927
658 XMLElement.SelectNodes Method 927
658.1 Syntax 927
658.2 Parameters 927
658.3 Return Value 927
658.4 Remarks 927
658.5 Code Example 928
659 ServerXMLHTTP.ResponseText Property 928
659.1 Access 928
659.2 Type 928
659.3 Remarks 928
659.4 Code Example 928
660 ServerXMLHTTP.ResponseXML Property 928
660.1 Access 929
660.2 Type 929
660.3 Remarks 929
660.4 Code Example 929
661 ServerXMLHTTP.ResponseBody Property 929
661.1 Access 929
661.2 Type 929
661.3 Remarks 930
661.4 Code Example 930
662 ServerXMLHTTP.Status Property 930
662.1 Access 930
662.2 Type 930
662.3 Remarks 930
662.4 Code Example 930
663 ServerXMLHTTP.StatusText Property 931
663.1 Access 931
663.2 Type 931
663.3 Remarks 931
663.4 Code Example 931

664 ServerXMLHTTP.ReadyState Property 931
664.1 Access 931
664.2 Type 932
664.3 Values 932
664.4 Remarks 932
664.5 Code Example 932
665 ServerXMLHTTP.Timeout Property 932
665.1 Access 933
665.2 Type 933
665.3 Default 933
665.4 Remarks 933
665.5 Code Example 933
666 DOMDocument.DocumentElement Property 933
666.1 Access 933
666.2 Type 933
666.3 Remarks 934
666.4 Code Example 934
667 DOMDocument.XML Property 934
667.1 Access 934
667.2 Type 934
667.3 Remarks 934
667.4 Code Example 934
668 DOMDocument.ParseError Property 935
668.1 Access 935
668.2 Type 935
668.3 Remarks 935
668.4 Code Example 935
669 DOMDocument.Async Property 935
669.1 Access 935
669.2 Type 935
669.3 Default 936
669.4 Remarks 936

669.5 Code Example 936
670 DOMDocument.ServerHTTPRequest Property 936
670.1 Access 936
670.2 Type 936
670.3 Default 936
670.4 Remarks 936
670.5 Code Example 937
671 DOMDocument.ResolveExternals Property 937
671.1 Access 937
671.2 Type 937
671.3 Default 937
671.4 Remarks 937
671.5 Code Example 937
672 DOMDocument.ValidateOnParse Property 937
672.1 Access 937
672.2 Type 938
672.3 Default 938
672.4 Remarks 938
672.5 Code Example 938
673 DOMDocument.PreserveWhiteSpace Property 938
673.1 Access 938
673.2 Type 938
673.3 Default 938
673.4 Remarks 938
673.5 Code Example 938
674 DOMDocument.SelectionLanguage Property 939
674.1 Access 939
674.2 Type 939
674.3 Default 939
674.4 Remarks 939
674.5 Code Example 939
675 DOMDocument.SelectionNamespaces Property 939

675.1 Access 939
675.2 Type 939
675.3 Remarks 940
675.4 Code Example 940
676 XMLNodeList.Length Property 940
676.1 Access 940
676.2 Type 940
676.3 Remarks 940
676.4 Code Example 940
677 XMLNodeList.Count Property 941
677.1 Access 941
677.2 Type 941
677.3 Remarks 941
677.4 Code Example 941
678 ParseError.ErrorCode Property 941
678.1 Access 941
678.2 Type 941
678.3 Remarks 941
678.4 Code Example 942
679 ParseError.Reason Property 942
679.1 Access 942
679.2 Type 942
679.3 Remarks 942
679.4 Code Example 942
680 ParseError.FilePos Property 942
680.1 Access 942
680.2 Type 943
680.3 Remarks 943
680.4 Code Example 943
681 ParseError.Line Property 943
681.1 Access 943
681.2 Type 943

681.3 Remarks 943
681.4 Code Example 943
682 ParseError.LinePos Property 944
682.1 Access 944
682.2 Type 944
682.3 Remarks 944
682.4 Code Example 944
683 ParseError.SrcText Property 944
683.1 Access 944
683.2 Type 944
683.3 Remarks 944
683.4 Code Example 944
684 ParseError.URL Property 945
684.1 Access 945
684.2 Type 945
684.3 Remarks 945
684.4 Code Example 945
685 XMLElement.NodeName Property 945
685.1 Access 945
685.2 Type 945
685.3 Remarks 946
685.4 Code Example 946
686 XMLElement.NodeValue Property 946
686.1 Access 946
686.2 Type 946
686.3 Remarks 946
686.4 Code Example 946
687 XMLElement.Text Property 947
687.1 Access 947
687.2 Type 947
687.3 Remarks 947
687.4 Code Example 947

688 XMLElement.XML Property 947
688.1 Access 947
688.2 Type 947
688.3 Remarks 947
688.4 Code Example 948
689 XMLElement.Attributes Property 948
689.1 Access 948
689.2 Type 948
689.3 Remarks 948
689.4 Code Example 948
690 XMLElement.ChildNodes Property 948
690.1 Access 948
690.2 Type 949
690.3 Remarks 949
690.4 Code Example 949
691 XMLElement.FirstChild Property 949
691.1 Access 949
691.2 Type 949
691.3 Remarks 949
691.4 Code Example 949
692 XMLElement.LastChild Property 950
692.1 Access 950
692.2 Type 950
692.3 Remarks 950
692.4 Code Example 950
693 XMLElement.ParentNode Property 950
693.1 Access 950
693.2 Type 950
693.3 Remarks 950
693.4 Code Example 951
694 XMLElement.Length Property 951
694.1 Access 951

694.2 Type 951
694.3 Remarks 951
694.4 Code Example 951
695 XMLElement.Children Property 951
695.1 Access 951
695.2 Type 952
695.3 Remarks 952
695.4 Code Example 952
696 ASP Example Applications 952
696.1 Overview 952
696.2 MVC — Model-View-Controller 952
696.3 MVVM — Model-View-ViewModel 953
696.4 REST — Basic REST-Style Endpoints 954
696.5 RESTful — Full RESTful API 954
696.6 Shared Patterns 955
696.7 Remarks 955
697 Database Convert Tool 955
697.1 Overview 955
697.2 Supported Conversions 955
697.3 How It Works 956
697.4 Prerequisites 956
697.5 Identifier Quoting 956
697.6 Type Mapping 957
697.7 Running the Tool 957
697.8 Remarks 957
698 Write New AxonASP Manual Pages 957
698.1 Overview 957
698.2 Syntax Template 957
698.3 Section Requirements 958
698.4 STRICT RULES & CONSTRAINTS (DO NOT IGNORE) 958
698.5 LLM Authoring Prompt 959
698.6 Example Code Guideline 960

699 Program Classic ASP with LLM/AI agents in AxonASP....961
699.1 Overview....961
699.2 Syntax....961
699.3 Parameters and Arguments....961
699.4 Return Values....962
699.5 Remarks....962
699.6 Agent Prompting Example....962
699.7 Code Example....965
700 Legal Disclaimer and License Information....965
700.1 Product Information....965
700.2 1. Software License....965
700.3 2. G3pix Trademarks and Logo Usage....973
700.4 3. Third-Party Trademarks and Affiliations....974
700.5 4. Disclaimer of Warranty....974
700.6 5. VBScript Package License....974
700.7 6. AxonASP Dependencies and Licenses....975

1 Welcome to AxonASP 2.0: The Ultimate Classic ASP Engine for the Modern Web

Supercharge your legacy code. Build blazing-fast modern APIs. Experience Classic ASP like never before.

Welcome to **AxonASP 2.0**, the definitive, high-performance runtime for executing Classic ASP and VBScript in pure Go. This is the engine your applications deserve—a complete reinvention of what Classic ASP can be in the modern era.

1.1 What Is AxonASP?

AxonASP is a high-performance execution engine that processes Classic ASP pages, compiles VBScript into optimized bytecode, and executes code through an advanced stack-based virtual machine. The engine provides complete compatibility with Microsoft Classic ASP standards while introducing powerful modern innovations that make ASP development faster, safer, and more scalable than ever before.

Unlike traditional ASP interpreters, AxonASP runs on **any operating system**—Windows, Linux, macOS, and beyond—using the same codebase. Deploy anywhere Go is supported. No more constraint to Windows and IIS.

1.2 Why Choose AxonASP? The Performance Revolution

We didn't just update Classic ASP. We completely reimagined it for the modern web:

1.2.1 Lightning-Fast Compilation and Execution

- **Zero AST, Pure Bytecode:** Single-pass compilation emits bytecode directly to a highly optimized stack-based Virtual Machine. By eliminating the Abstract Syntax Tree (AST), AxonASP executes scripts with virtually zero-allocation overhead. It's insanely fast and memory efficient.
- **Advanced Caching Architecture:** IIS-style VM pooling combined with aggressive script caching and dynamic execution caching. Your compiled code is cached at every level—script files, Eval expressions, Execute statements, and beyond.
- **Minimal Memory Footprint:** Code is engineered to minimize garbage collection triggers. Scripts complete faster and consume less memory than traditional interpreters.

1.2.2 *Total Compatibility Meets Innovation*

- **Classic ASP Standards Compliance:** Your legacy code drops right in. Full support for VBScript semantics, ADOB, MSXML, FileSystemObject, Dictionary, WScript.Shell, and all intrinsic ASP objects.
- **60+ Custom Axon Functions:** High-performance native functions for arrays, strings, system operations, and advanced utilities—written in Go, executing at near-native speed.
- **Enterprise Native Libraries:** G3JSON, G3DB, G3HTTP, G3Mail, G3Image, G3Crypto, G3ZIP, G3TAR, G3ZSTD, G3PDF, G3Template, and many more—zero overhead, maximum power.

1.2.3 *Multiple Execution Modes*

- **Web Server Mode:** Run the built-in HTTP server (`axonasp-http`) for development and production.
- **FastCGI Mode:** Integrate with Nginx, Apache, or any FastCGI-compatible web server.
- **Command-Line Mode:** Execute ASP scripts directly from your terminal (`axonasp-cli`). Schedule tasks, automation, system administration—all in ASP.
- **Interactive CLI with TUI:** Explore and test code in real-time with a text-based user interface.

1.2.4 *AI-Ready Architecture*

- **Built-in MCP Server:** AxonASP includes a Model Context Protocol (MCP) server. AI agents can connect directly, understand your environment, and autonomously author complete ASP pages using all available native functions.

1.2.5 *Testing and Quality Assurance*

- **Automated Test Suite:** The `axonasp-testsuite` executable enables test-driven development natively in ASP. Write assertions, run test discovery automatically, and report CI-friendly results.
- **No More Regex Hacks:** Integrated testing framework—no more manual test pages and prayer-based debugging.

1.3 What's New in Version 2.0?

- **Unified Configuration:** Centralized `axonasp.toml` configuration with `.env` support via Viper. Single source of truth for all settings.
- **Modern Architecture Examples:** Complete, production-ready examples for REST, RESTful, MVC, and MVVM—all written in pure ASP.

- **Comprehensive Local Documentation:** The complete manual is built right into the repository. No need to hunt through old forums or outdated blogs. Everything you need is in `./www/manual/md/`.
- **Intelligent Port Defaults:** Updated default proxy port (8801) to avoid firewall conflicts and system port contention out of the box.
- **Native Docker Support:** Containerization with included `Dockerfile` and `docker-compose.yml`. Deploy in seconds.
- **Database Migration Tool:** Built-in converter to migrate legacy Access databases to modern formats (SQLite, MySQL, PostgreSQL, MSSQL).

1.4 Understanding This Manual

This manual is organized to help you understand AxonASP from the ground up:

1.4.1 Section: Runtime

Covers deployment modes, internal architecture, script caching, locale support, global.asa lifecycle, web.config directives, FastCGI setup, reverse proxy configuration, running as a Linux service, installing from pre-built Linux packages (DEB, RPM, APK), the service wrapper, system and error pages, and MCP Server integration with VS Code.

1.4.2 Section: Configuration

Documents the `axonasp.toml` configuration file: all available keys, default values, and explanations for each setting that controls the engine, server, sessions, caching, and logging.

1.4.3 Section: ASP Foundations

A complete reference for Classic ASP and VBScript fundamentals: the `#include` directive, variables, procedures, conditionals, looping, syntax overview, quick reference, forms, cookies, and the five intrinsic ASP objects (Request, Response, Server, Session, Application), plus ASPError and Scripting.Dictionary.

1.4.4 Section: Libraries

Full API reference for every built-in native library and compatibility object:

Native AxonASP Libraries G3AXON Functions, G3CRYPTO, G3JSON, G3DB, G3HTTP, G3MAIL, G3IMAGE, G3FILES, G3TESTSUITE, G3TEMPLATE, G3ZIP, G3ZLIB, G3TAR, G3ZSTD, G3FC, G3MD, G3PDF, G3FILEUPLOADER.

Compatibility Objects WScript.Shell, ADOX.Catalog, MSWC Components (AdRotator, NextLink, ContentRotator, Counters, Tools, PageCounter, BrowserType,

PermissionChecker, MyInfo), ADODB Family (Connection, Recordset, Command, Fields, Parameters, Errors), ADODB.Stream, Scripting.Dictionary, Scripting.FileSystemObject, VBScript.RegExp, MSXML2 Family (ServerXMLHTTP, DOMDocument, XMLNodeList, XMLElement).

1.4.5 Section: Examples

Working code demonstrating architectural patterns, library usage, and best practices for Classic ASP applications built on AxonASP.

1.4.6 Section: Tools

Documentation for the built-in database migration tool, which converts legacy Access databases to modern formats including SQLite, MySQL, PostgreSQL, and MSSQL.

1.4.7 Section: Authoring

Guidelines for writing manual pages and for programming Classic ASP with LLMs. Read these before contributing documentation.

1.5 Key Capabilities at a Glance

Server-Side Power: - Classic ASP page execution with 99% Microsoft compatibility - VBScript with all standard functions and operators - Full intrinsic object support (Request, Response, Server, Session, Application) - Advanced error handling with line-level debugging

Native Libraries (Zero-Overhead): G3JSON, G3DB, G3HTTP, G3Mail, G3Image, G3Crypto, G3Zip, G3TAR, G3ZSTD, G3ZLIB, G3Template, G3PDF, G3MD, G3FC, G3FileUploader, and G3TestSuite.

Compatibility Objects: ADODB (with real database backends), MSXML, Scripting.FileSystemObject, Scripting.Dictionary, WScript.Shell, VBScript.RegExp, and MSWC components.

Production Ready: Reverse proxy mode, FastCGI integration, command-line execution, automated testing, MCP for AI agents, comprehensive logging, and performance monitoring.

1.6 Getting Started

To begin working with AxonASP, start by reading about the runtime architecture and deployment options. Then select the deployment model that fits your infrastructure, configure your environment, and deploy your applications with confidence.

For detailed API reference on any library or object, consult the menu on the left or use `./www/manual/menu.md` for a complete navigational structure.

1.7 A Word About Version 2.0

Version 1.0 of AxonASP is completely deprecated and incompatible with Version 2.0. We made deliberate architectural decisions to deliver superior performance, maintainability, and feature parity with modern requirements. The investment in rewriting the engine from the ground up pays dividends in speed, reliability, and joy of development.

If you're upgrading from Version 1.0, your AxonASP code and configuration *will require migration*. It's worth the effort—the improvements are transformational.

1.8 Questions? Ready to Deploy?

This manual contains everything you need to understand, configure, and deploy AxonASP. Each section builds on prior knowledge, so start at the beginning if you're new, or jump directly to specific topics you need to solve.

Your Classic ASP applications—and your team—are about to experience what the platform can truly do.

2 Understand the AxonASP Project Structure

2.1 Overview

AxonASP is a Go-based Classic ASP and VBScript execution runtime that provides multiple deployment modes. The project contains a shared VM/compiler core and multiple executable entry points for different use cases. All executables are built using the **build.ps1** (Windows) or **build.sh** (Linux/macOS) script and placed in the project root directory.

2.2 Building

Windows (PowerShell):

```
.\build.ps1
```

Linux / macOS (Bash):

```
./build.sh
```

For Linux cross-compilation to a specific architecture:

```
./build.sh --platform linux --arch amd64
./build.sh --platform linux --arch arm64
```

Both scripts build the following executables:

Executable	Description
`axonasp-http` / `axonasp-http.exe`	HTTP/HTTPS web server (port 8801)
`axonasp-fastcgi` / `axonasp-fastcgi.exe`	FastCGI application server (port 9000)
`axonasp-cli` / `axonasp-cli.exe`	Command-line interpreter and TUI
`axonasp-mcp` / `axonasp-mcp.exe`	Model Context Protocol server for AI integration
`axonasp-testsuite` / `axonasp-testsuite.exe`	Automated ASP test suite runner

2.3 Project Directory Structure

```
axonasp2/
├── axonvm/                    # Core VM, compiler, and intrinsic objects
│   ├── asp/                   # ASP intrinsic objects (Request, Response,
 Session, etc.)
│   ├── lib_*.go               # Native object libraries (ADODB, MSXML, F
SO, G3*, etc.)
│   ├── compiler*.go           # Single-pass compiler emitting bytecode
│   ├── vm.go                  # Stack-based virtual machine execution
│   ├── opcode.go              # VM opcodes and bytecode definitions
│   └── value.go               # VM value type system
│
├── vbscript/                  # Lexical analyzer and VBScript parser
│   ├── lexer.go               # Token generation from ASP source
│   ├── parser.go              # ASP parsing and validation
│   ├── token.go               # Token definitions
│   └── vberrorcodes.go        # VBScript error numbers/messages
│
├── server/                    # HTTP web server runtime
│   ├── main.go                # HTTP listener and request handler
│   ├── web_host.go            # ASP execution and routing
│   ├── webconfig.go           # web.config parsing and rules
│   └── directorylisting.go    # Directory listing UI generation
│
├── fastcgi/                   # FastCGI application server runtime
```

```
│   └── main.go                # FastCGI listener and protocol handler
│
├── cli/                       # Command-line interface runtime
│   └── main.go                # TUI and script execution
│
├── mcp/                       # Model Context Protocol server
│   └── main.go                # MCP stdio/SSE server
│
├── testsuite/                 # Automated ASP test suite runner
│   └── main.go                # Test runner entry point
│
├── service/                   # Background service runtime
│   └── main.go                # Service entry point
│
├── config/
│   └── axonasp.toml           # Configuration file (shared by all runtim
es)
│
├── axonconfig/
│   └── loader.go              # Configuration loader (shared by all runt
imes)
│
├── www/                       # Web root directory
│   ├── manual/                # Built-in documentation
│   ├── tests/                 # ASP test suite pages
│   ├── error-pages/           # Custom HTTP error page templates
│   ├── axonasp-pages/         # System pages, CSS, and assets
│   ├── database-convert/      # Access database conversion tool
│   ├── mvc/                   # MVC example application
│   ├── mvvm/                  # MVVM example application
│   ├── rest/                  # REST example application
│   └── restful/               # RESTful example application
│
├── resources/                 # Static resources and data
├── docker/                    # Docker deployment files
├── temp/                      # Runtime cache and session storage
├── global.asa                 # Application-level event handlers
├── build.ps1                  # PowerShell build script (Windows)
├── build.sh                   # Bash build script (Linux/macOS)
└── go.mod, go.sum             # Go module dependencies
```

2.4 Required Runtime Files

When deploying AxonASP executables, ensure the following are in the **same directory** as the executable:

1. **config/axonasp.toml** - Configuration file (absolute or relative path, configurable)
2. **www/** - Web root directory containing your ASP applications

Both paths are configured in `axonasp.toml` and can be set via environment variables:

```
$env:WEB_ROOT = "C:\myapp\www"
$env:CONFIG_PATH = "C:\myapp\config\axonasp.toml"
```

2.5 Environment Variable Override

All configuration values in `axonasp.toml` can be overridden via **environment variables** (requires `viper_automatic_env = true` in config):

```
# Format: SECTION_SETTING or SECTION_SUBSETTING (uppercase, underscores r
eplace dots)
$env:DEFAULT_CHARSET = "UTF-8"
$env:DEFAULT_SCRIPT_TIMEOUT = "120"
$env:SERVER_PORT = "8802"
```

2.6 Service Ports and Endpoints

Service	Executable	Default Port	Purpose
HTTP	axonasp-http.exe / axonasp-http	8801	Direct web server (development/ proxy backend)
FastCGI	axonasp-fastcgi.exe / axonasp-fastcgi	9000	FastCGI application server
CLI	axonasp-cli.exe / axonasp-cli	N/A	Command-line script execution
MCP	axonasp-mcp.exe / axonasp-mcp	stdio / SSE	AI model integration server
Test Suite	axonasp-testsuite.exe / axonasp-testsuite	N/A	Automated ASP test runner

2.7 Docker Support

AxonASP includes Docker deployment support via **docker-compose.yml** and **Dockerfile**. Build and run containerized instances:

```
docker-compose up -d
```

This creates isolated, easily scalable AxonASP instances suitable for cloud deployment or load-balancing scenarios.

2.8 Deployment Architecture

2.8.1 NOT Recommended: Direct Web Exposure

❌ **DO NOT** expose axonasp-http.exe directly to public internet traffic - Increases security surface area - No centralized TLS/SSL termination - No rate-limiting or DDoS protection - No request logging/monitoring integration

2.8.2 Recommended: Reverse Proxy Mode (axonasp-http)

Run AxonASP behind a reverse proxy (Nginx, Apache, IIS, Caddy) for security, TLS offloading, and load-balancing.

Benefits: - Centralized authentication and authorization - TLS/SSL certificate management - Request rate-limiting and throttling - Web Application Firewall (WAF) integration - Multiple backend instances for redundancy - Static asset caching and compression

2.8.3 Recommended: FastCGI Mode (axonasp-fastcgi)

Use FastCGI protocol for direct integration with other web servers without reverse proxy overhead.

Benefits: - Native protocol support in Nginx, Apache, IIS - Lower latency than proxy forwarding - Direct request handling with minimal translation - Better integration with native web server features

---

2.9 Deployment Architecture

AxonASP should not be exposed directly to public internet traffic. The recommended deployment patterns are:

- **Reverse Proxy Mode** — Run `axonasp-http` behind Nginx, Apache, or Caddy. The proxy handles TLS, rate-limiting, and static content. AxonASP handles only ASP execution.
- **FastCGI Mode** — Use `axonasp-fastcgi` as a backend process integrated directly with Nginx, Apache, or IIS using the FastCGI protocol.
- **Linux Service** — Run any executable as a persistent `systemd` service for production Linux deployments.

For detailed configuration examples see:

- Reverse Proxy Setup
- FastCGI Setup
- Running as a Linux Service

- MCP Server and VS Code Integration

2.10 Configuration Reference

For detailed information about each configuration option, see the axonasp.toml documentation.

2.11 Remarks

- **HTTP Server** (`axonasp-http`): High-performance web server suitable as a proxy backend or development server.
- **FastCGI Server** (`axonasp-fastcgi`): Lightweight FastCGI handler for native web server integration.
- **CLI** (`axonasp-cli`): Development and maintenance tool with interactive TUI and batch script support.
- **MCP Server** (`axonasp-mcp`): AI model integration for code generation, analysis, and documentation.
- **Test Suite** (`axonasp-testsuite`): Automated runner that discovers and executes ASP test files using the `G3TestSuite` object.
- Runtime feature parity is maintained across all deployment modes.
- All runtimes share a single `config/axonasp.toml` configuration file.
- On Linux and macOS, executables are named without the `.exe` extension.

3 Understand AxonASP Internal Architecture (Go VM and Compiler)

3.1 Overview

AxonASP is a modern execution engine for Classic ASP and VBScript, implemented in Go and designed for high compatibility, high throughput, and long-term maintainability.

This document explains how AxonASP works internally so that developers and automation agents can implement new runtime features, native libraries, and compatibility behaviors with confidence.

3.2 Origin Story and Mission

AxonASP started from a practical risk: once VBScript deprecation was announced by Microsoft, long-term support for Classic ASP became uncertain.

The project mission is straightforward: - Keep Classic ASP alive for production workloads. - Preserve historical behavior where compatibility matters. - Evolve the platform with modern capabilities without breaking legacy applications.

AxonASP modernizes the ecosystem by providing: - Cross-platform execution on Windows, Linux, and macOS. - Multiple runtime modes: HTTP server, FastCGI server, CLI interpreter, and MCP server. - Modern developer workflows through CLI and TUI-style interactive execution, plus MCP integration for model-assisted development and web application maintenance.

3.3 Core Engine Design

3.4 High-Level Pipeline

AxonASP executes ASP/VBScript through a direct lexer-to-bytecode pipeline:

1. Source is tokenized by the lexer in `vbscript/`.
2. Compiler consumes tokens and emits VM opcodes directly.
3. Stack-based VM executes bytecode against a typed `Value` model.
4. Intrinsic ASP objects and native libraries are dispatched through VM-native object routing.

This architecture removes AST construction overhead and minimizes intermediate allocations.

3.5 Runtime Modes Share One Core

The following executables reuse the same compiler/VM core in `axonvm/`: - HTTP server (`server/`) - FastCGI server (`fastcgi/`) - CLI (`cli/`) - MCP server (`mcp/`)

Feature parity is expected across all modes.

3.6 Go Implementation Strategy and Optimizations

3.7 Wrapping Go Libraries as ASP Objects

AxonASP extends Classic ASP by wrapping Go implementations behind `Server.CreateObject(...)` compatibility objects.

Typical flow: - A ProgID (for example, `MSXML2.DOMDocument` or a custom library ProgID) is passed to `Server.CreateObject`. - VM creates a concrete Go struct instance for that object. - VM assigns a dynamic native ID and returns a `VTNativeObject` value. - Member access is routed through typed dispatch functions, not reflection.

This allows ASP pages to consume modern Go functionality through object semantics that feel native to Classic ASP.

3.8 Performance Discipline in Go

AxonASP prioritizes low-overhead execution: - Avoid `interface{}`-centric runtime paths in VM hot loops. - Avoid reflection (`reflect`) for object routing. - Prefer explicit typed dispatch (`switch` / `strings.EqualFold`) for member resolution. - Keep data in compact structs and primitive fields where possible. - Reduce temporary allocations and unnecessary conversions in compile/execute phases.

The VM value model (`Value`) is a tagged struct that avoids generic boxed object payloads in the execution path.

3.9 Compiler Architecture (No AST Rule)

3.10 Rule: No AST

AxonASP does not build an Abstract Syntax Tree for script compilation.

Instead, the compiler performs immediate bytecode emission while reading tokens. This improves: - Compilation latency - Memory profile - Throughput under dynamic script generation workloads

3.11 What Replaces AST

The compiler uses: - Token stream parsing with direct opcode emission. - Constant pool indexing for names/literals. - Jump patching for control-flow targets. - Class/member registration opcodes for runtime metadata. - Context-aware emission for VBScript compatibility semantics.

Representative behavior in compiler internals: - Direct expression emission (`OpAdd`, `OpConcat`, comparisons, logical ops). - Value-context coercion (`OpCoerceToValue`) where VBScript semantics require default-property reads. - ByRef patching with argument reference opcodes (`OpArgGlobalRef`, `OpArgLocalRef`, `OpArgClassMemberRef`). - Class/property registration opcodes (`OpRegisterClass*`) consumed by VM at runtime.

3.12 Compiler Tooling for Runtime Efficiency

Compiler and execution tooling includes: - Directive compilation support for `<%@ ... %>` blocks. - Dynamic expression execution support (`Eval` / dynamic compile paths). - Eval bytecode caching (LRU) keyed by expression and compare-mode/scope signatures. - Script cache layers for reducing repeated compilation work.

The guiding pattern is stable bytecode reuse whenever scope and options permit.

3.13 VM Architecture

3.14 VM Type

AxonASP VM is: - Stack-based - Bytecode-driven - Tagged-value runtime

Static stack capacity is fixed (`StackSize = 4096`) and execution is opcode-oriented.

3.15 Opcode Model

The opcode set is organized by category: - Data movement (`OpGetGlobal`, `OpSetLocal`, etc.) - Arithmetic and logical operations - Control flow and jumps - Output writing (`OpWrite`, `OpWriteStatic`) - Class registration and instantiation - Member calls and dispatch (`OpCallMember`, `OpMemberGet`, `OpMemberSet`) - Error mode handling (`OpOnErrorResumeNext`, `OpOnErrorGoto0`)

This keeps runtime deterministic and predictable for compatibility-critical behaviors.

3.16 Call Frames and ByRef

User-defined Sub/Function calls create VM call frames that track: - Return instruction pointer - Frame pointers and stack restoration - Bound object for class-member context - ByRef write-back mapping to caller slots - Error mode state restoration

This enables VBScript-compatible ByRef semantics without dynamic reflection.

3.17 Intrinsic ASP Object Emulation

AxonASP pre-reserves intrinsic object slots and exposes them as native VM objects: - `Response` - `Request` - `Server` - `Session` - `Application` - `ObjectContext` - `Err`

These are available through the ASP execution environment and participate in normal member dispatch paths.

State behavior: - Session state persists under `temp/session` with ASP session cookies. - Application state is process-resident memory.

Error information is bridged to the ASP error surface (`ASPError`) with line, column, file, and categorized runtime diagnostics.

3.18 Type System and Memory Model

3.19 Variant Representation in Go

VBScript dynamic values are represented by `axonvm.Value`: - Type tag (`VTEmpty`, `VTNull`, `VTBool`, `VTInteger`, `VTDouble`, `VTString`, `VTDate`, `VTArray`, `VTObject`, `VTNativeObject`, `VTBuiltin`, `VTUserSub`, `VTArgRef`) - Numeric fields (`Num`, `Flt`) - String payload (`Str`) - Array pointer (`Arr`) - Name metadata (`Names`) for subroutines/object fields

This design supports: - Fast type checks in hot opcodes - Compatibility coercions - Reference-aware call behavior (ByRef) - Minimal object-shape ambiguity at runtime

3.20 Compatibility Semantics

Core compatibility rules preserved by compiler + VM cooperation include: - Case-insensitive identifiers/member names - 1-based behaviors where required by VBScript semantics - Property Get/Let/Set dispatch distinctions - Object/value coercion behavior - `On Error Resume Next` and recovery behavior

---

3.21 Native Object and Library Lifecycle

3.22 Creation

On `Server.CreateObject(progID)`: 1. VM normalizes ProgID for compatibility matching. 2. VM allocates concrete Go object instance. 3. VM assigns dynamic native ID. 4. Instance is stored in a VM-owned map for that library type. 5. Return value is `Value{Type: VTNativeObject, Num: dynamicID}`.

3.23 Dispatch

Member calls are routed by VM through: - Native ID range/type map resolution - Library-specific `DispatchMethod` and `DispatchPropertyGet`/`DispatchPropertySet` logic - Argument conversion and return value conversion using `Value` helpers

3.24 Cleanup and Lifetime

Object lifetimes are tied to VM/request execution context and reference behavior. Deterministic cleanup patterns and map lifecycle management are critical to avoid retained objects after script completion.

3.25 Practical Extension Guide

3.26 Add a New Native Library (Recommended Pattern)

1. Create `axonvm/lib_<name>.go` with a concrete struct.
2. Implement strongly typed dispatch members:
    - `DispatchMethod(methodName string, args []Value) Value`
    - `DispatchPropertyGet(propertyName string) Value`
    - Optional property set handler when writable members exist.
3. Register object map in VM state.
4. Integrate ProgID creation in VM native call routing.
5. Route method/property dispatch for that dynamic ID space.
6. Use `Value` constructors (`NewString`, `NewInteger`, `NewBool`, etc.) consistently.
7. Raise standardized runtime errors for invalid args/state instead of silent `Empty` on operational failures.

3.27 Add Compiler/VM Features Safely

When adding language features or compatibility behavior: - Preserve no-AST direct emission architecture. - Add opcodes only when necessary and wire full lifecycle (emit + execute + error handling). - Keep compatibility semantics explicit in compiler emission rules. - Verify parity in HTTP, FastCGI, CLI, and MCP execution paths. - Add regression tests for compile and runtime behavior.

3.28 Internal Files to Know First

Start here for architecture work: - `axonvm/vm.go` - VM runtime, call frames, native object routing, intrinsic object plumbing. - `axonvm/opcode.go` - Opcode definitions and instruction taxonomy. - `axonvm/value.go` - Variant representation and value helpers. - `axonvm/compiler*.go` - Token-to-bytecode direct emission logic. - `vbscript/` - Lexer and VBScript token definitions/errors. - `axonvm/lib_*.go` - Native library implementations.

3.29 Design Principles Summary

AxonASP is built around four non-negotiable principles: - Compatibility first for Classic ASP/VBScript behavior. - Single-pass no-AST compilation for speed and memory efficiency. - Stack-based bytecode VM with strict typed-value runtime. - Native Go extensions exposed as ASP objects without reflection-heavy dispatch.

If you follow these principles, you can extend AxonASP quickly while preserving both performance and compatibility.

4 Use axonasp-testsuite

4.1 Overview

`axonasp-testsuite` is the native Classic ASP test runner for AxonASP. Conceptually, it serves the same role as PHPUnit for PHP projects or `go test` for Go projects: discover tests, execute them, aggregate assertions, print a human-friendly summary, and return a process exit code suitable for CI/CD pipelines.

The runner executes test pages through the AxonASP VM, collects assertion reports from each `G3Test` or `G3TestSuite` object created in each file, and fails the process when any suite has failed assertions, runtime errors, compile errors, or no assertions.

4.2 Discovery Rules

The runner recursively scans a directory and executes files that match:

- `*test.asp`
- `test_*.asp`

The default extension list comes from `global.execute_as_asp` in `config/axonasp.toml`. Files are sorted before execution to provide deterministic run order.

4.3 Command-Line Usage

Windows:

```
.\axonasp-testsuite.exe .\www\tests\testsuite
```

Linux and macOS:

```
./axonasp-testsuite ./www/tests/testsuite
```

You can run a broader folder:

```
.\axonasp-testsuite.exe .\www\tests
```

4.4 How Test Files Are Executed

For each discovered file, the runner performs this flow:

1. Loads or compiles the ASP script through the bytecode cache.
2. Creates a VM instance from the cached program.

3. Sets a CLI-style ASP host context (`REQUEST_METHOD=CLI`) with server root and request path.
4. Executes the file and captures output and assertion reports.
5. Prints per-file status and aggregate summary.

A file is marked as failed when any of the following happens:

- Compile error
- Runtime error
- One or more failed assertions
- No assertions executed (no `G3Test`/`G3TestSuite` report generated)

4.5 Creating ASP Tests

Inside each test file, instantiate the test object with:

```
Set t = Server.CreateObject("G3TestSuite")
```

`"G3Test"` is also accepted as an alias:

```
Set t = Server.CreateObject("G3Test")
```

4.5.1 Minimal Smoke Test

```
<%
Option Explicit

Dim t
Set t = Server.CreateObject("G3TestSuite")

t.Describe "G3TestSuite smoke test"
t.AssertEqual 4, 2 + 2, "2 + 2 should equal 4"
t.AssertTrue Len("axon") = 4, "Len should report the expected size"
t.AssertFalse IsEmpty("ready"), "Literal text should not be Empty"
%>
```

4.6 Assertion API

The following methods are supported by the native `G3Test` object.

4.6.1 Describe

Sets the active suite/block label for failure messages.

```
t.Describe "Math operations"
```

4.6.2 AssertEqual(expected, actual, [message])

Passes when values are equal under VM comparison/coercion semantics.

4.6.3 AssertNotEqual(expected, actual, [message])

Passes when values are different.

4.6.4 AssertTrue(condition, [message])

Passes when condition is truthy under VBScript boolean coercion.

4.6.5 AssertFalse(condition, [message])

Passes when condition is falsy under VBScript boolean coercion.

4.6.6 AssertEmpty(value, [message])

Passes when value is VBScript `Empty`.

4.6.7 AssertNull(value, [message])

Passes when value is VBScript `Null`.

4.6.8 AssertNothing(value, [message])

Passes when value is `Nothing`-compatible in AxonASP semantics.

4.6.9 AssertTypeName(expectedType, value, [message])

Passes when `TypeName(value)` matches `expectedType` (case-insensitive).

4.6.10 AssertLength(expectedLength, value, [message])

Passes when the target value length/count equals expected length.

Supported targets include:

- Strings (character length)
- Arrays (element count)
- `Empty` (treated as length 0)
- Native objects exposing `Count`

4.6.11 AssertCount(expectedCount, value, [message])

Alias of `AssertLength`.

4.6.12 AssertRaises(code, [expected], [message])

Executes dynamic VBScript code and verifies an error is raised.

Supported argument forms:

- `AssertRaises(code, message)`
- `AssertRaises(code, expectedNumber, message)`

- `AssertRaises(code, expectedText, message)`

Examples:

```
t.AssertRaises "Err.Raise 13, \"suite\", \"type mismatch\"", 13, "Should
raise Err.Number 13"
t.AssertRaises "Err.Raise 13, \"suite\", \"type mismatch\"", "type mismat
ch", "Should contain error text"
t.AssertRaises "Function Broken(", "Syntax errors should be trapped"
```

4.6.13 Fail([message])

Forces an explicit failure.

```
t.Fail "This branch should be unreachable"
```

4.7 Suite Properties

These read properties are available:

- `Suite` / `Description` / `CurrentDescribe`
- `Total` / `TotalTests`
- `Passed`
- `Failed`
- `HasFailures`

The suite name can also be set through writable aliases:

```
t.Suite = "Authentication"
```

4.8 Practical Test Organization

- Keep one feature area per file (`auth_test.asp`, `json_test.asp`).
- Start each block with `Describe` to make failure output easy to scan.
- Use explicit assertion messages that describe intent, not implementation.
- Prefer deterministic data; avoid dependence on clock/timezone unless explicitly testing those behaviors.
- Keep one assertion intent per line so failed output maps directly to a single behavior.

4.9 Example: Multi-Assertion Test File

```
<%
Option Explicit

Dim t, arr
Set t = Server.CreateObject("G3TestSuite")

t.Describe "Core VBScript compatibility"
```

```
t.AssertEqual 10, CLng(9.6), "CLng should round using VBScript rules"
t.AssertTypeName "String", CStr(123), "CStr should produce String"
t.AssertNotEqual "abc", "xyz", "Different strings must not be equal"

arr = Array("a", "b", "c")
t.AssertCount 3, arr, "Array should expose element count"

t.Describe "Error behavior"
t.AssertRaises "Err.Raise 5, \"suite\", \"invalid procedure call\"", 5, "
Err.Raise number should match"
%>
```

4.10 CI Usage

Example CI step:

```
- name: Build axonasp-testsuite
  run: go build -o axonasp-testsuite ./testsuite

- name: Run ASP test suite
  run: ./axonasp-testsuite ./www/tests/testsuite
```

Use process exit code:

- `0` when all suites pass
- `1` when any suite fails or when execution is invalid

4.11 Runtime Integration Notes

- `global.asa` is loaded before suite execution and application/session end hooks are called when the run finishes.
- The runner uses the shared VM pool and script cache for fast execution.
- Output includes per-file PASS/FAIL status, assertion counters, and detailed failure messages.
- If a test file writes to output, that output is displayed under the file report for diagnostics.

4.12 Remarks

- Treat `axonasp-testsuite` as your authoritative regression gate for ASP behavior changes.
- Prefer `G3TestSuite` in documentation and project examples; use `G3Test` alias only when needed.
- Ensure every test file executes at least one assertion; files with no assertions are treated as failure by design.

5 global.asa

5.1 Overview

`global.asa` is the application-level script file that defines Application and Session lifecycle event handlers and declares static objects for the `Application` and `Session` scopes. AxonASP compiles `global.asa` once at startup and runs the appropriate handlers when application or session events fire.

5.2 File Location

Place `global.asa` at the root of the configured web root directory, alongside your default `default.asp` or `index.asp` file. Both the HTTP server and the FastCGI server look for the file in the configured web root.

When using the CLI (`axonasp-cli.exe`), `global.asa` must be placed in the **same directory as the CLI executable**. The CLI does not use the web root path for this file.

5.3 Supported Event Handlers

Declare standard Sub procedures using the following exact names inside a `<SCRIPT LANGUAGE="VBScript" RUNAT="Server">` block:

Event	Fired When
Application_OnStart	The application starts on the first request after server startup
Application_OnEnd	The application ends on clean server shutdown
Session_OnStart	A new session is created for a client
Session_OnEnd	A session expires or is abandoned

```
<SCRIPT LANGUAGE="VBScript" RUNAT="Server">

Sub Application_OnStart()
    Application("SiteTitle") = "My AxonASP Site"
    Application("StartTime") = Now()
End Sub

Sub Application_OnEnd()
    ' Clean up application-level resources
End Sub

Sub Session_OnStart()
    Session("CreatedAt") = Now()
    Session("RequestCount") = 0
End Sub
```

```
Sub Session_OnEnd()
    ' Clean up session-level resources
End Sub

</SCRIPT>
```

5.4 Static Object Declarations

Declare objects with Application or Session scope using the `<OBJECT>` tag. Static objects are automatically available throughout the application without calling `Server.CreateObject` on each page.

```
<OBJECT RUNAT="Server" SCOPE="Application" ID="AppCache" PROGID="G3JSON">
</OBJECT>

<OBJECT RUNAT="Server" SCOPE="Session" ID="UserData" PROGID="G3JSON">
</OBJECT>
```

OBJECT tag attributes:

Attribute	Values	Description
RUNAT	Server	Required. Must always be Server
SCOPE	Application / Session	Lifetime and sharing scope of the object
ID	identifier	Variable name used to access the object in ASP pages
PROGID	ProgID string	Object class identifier (same as the string passed to Server.CreateObject)
CLASSID	CLSID string	Alternative to PROGID for COM class identifiers

5.5 What Is Not Supported

The following Classic ASP `global.asa` features are not supported in AxonASP:

- **TypeLib declarations** — `<METADATA TYPE="TypeLib" ...>` directives are ignored.
- **Request and Response access in Application events** — `Request` and `Response` objects are not available inside `Application_OnStart` and `Application_OnEnd`. The server suppresses all response output from `global.asa` handlers to match IIS behavior.
- **#include directives** — File includes are not processed inside `global.asa`.

- **ObjectContext** — The `ObjectContext` intrinsic object is not available in event handlers.
- **On Error Resume Next suppressing compile errors** — A compile error in `global.asa` aborts server startup.

5.6 Remarks

- `global.asa` is compiled once when the server starts. Restart the server to pick up changes to the file.
- UTF-8 BOM is automatically stripped from `global.asa` before compilation.
- Application and Session scope static objects declared in `global.asa` are initialized lazily when first accessed by an ASP page.
- `Session_OnEnd` fires when a session expires based on the configured timeout, not necessarily when the user closes the browser.
- `Application_OnEnd` is only fired during a clean server shutdown. An OS-level process kill will not trigger this event.
- The CLI uses the `global.asa` file located in the same directory as `axonasp-cli.exe`, not from a web root path.
- All four event handlers are optional. You can define only the events your application needs.

5.7 Code Example

A typical `global.asa` that initializes a shared counter and tracks session creation time:

```
<SCRIPT LANGUAGE="VBScript" RUNAT="Server">

Sub Application_OnStart()
    Application("TotalVisits") = 0
    Application("AppStarted") = Now()
End Sub

Sub Session_OnStart()
    Application.Lock
    Application("TotalVisits") = Application("TotalVisits") + 1
    Application.Unlock
    Session("SessionStart") = Now()
End Sub

Sub Session_OnEnd()
    ' Nothing to clean up
End Sub

Sub Application_OnEnd()
    ' Nothing to clean up
```

```
End Sub

</SCRIPT>
```

6 web.config Support

6.1 Overview

AxonASP HTTP server reads a single `web.config` file placed at the **root** of the configured web application directory and applies URL rewrite rules, HTTP redirects, and custom error page mappings. The file follows a subset of the IIS XML schema under `<system.webServer>`.

Only the web root `web.config` is read. Files placed in subdirectories are ignored.

6.2 Prerequisites

Enable web.config processing in `config/axonasp.toml`:

```
[server]
enable_webconfig = true
```

When `enable_webconfig` is `false`, the server ignores all `web.config` files and uses only the settings from `axonasp.toml`.

6.3 Supported Sections

6.3.1 URL Rewrite Rules

Declare rules inside `<system.webServer><rewrite><rules>`. Each rule is evaluated in document order against the incoming request path.

```
<configuration>
  <system.webServer>
    <rewrite>
      <rules>
        <rule name="Front Controller" stopProcessing="true">
          <match url="^(?!index\.asp$)(.*)" ignoreCase="true" />
          <conditions logicalGrouping="MatchAll">
            <add input="{REQUEST_FILENAME}" matchType="IsFile" negate="tr
ue" />
            <add input="{REQUEST_FILENAME}" matchType="IsDirectory" negat
e="true" />
          </conditions>
          <action type="Rewrite" url="/index.asp?route={R:1}" appendQuery
String="true" />
        </rule>
      </rules>
    </rewrite>
```

```
  </system.webServer>
</configuration>
```

Rule element attributes:

Attribute	Values	Default	Description
name	string	—	Display name for the rule
stopProcessing	true / false	false	Stop evaluating further rules when this rule matches

Match element attributes:

Attribute	Values	Default	Description
url	regex	—	URL pattern matched against the request path without the leading slash
ignoreCase	true / false	true	Perform case-insensitive matching
negate	true / false	false	Invert the match result

Conditions element attributes:

Attribute	Values	Default	Description
logicalGrouping	MatchAll / MatchAny	MatchAll	Apply AND or OR logic to all conditions in the block

Condition add element attributes:

Attribute	Values	Default	Description
input	see below	—	Server variable or expression to evaluate
matchType	IsFile / IsDirectory / Pattern	Pattern	How the input is evaluated
pattern	regex	—	Regular expression pattern when matchType is Pattern
ignoreCase	true / false	true	Case-insensitive regex matching
negate	true / false	false	Invert the condition

Attribute	Values	Default	Description
			result

Supported condition input variables:

Variable	Description
{REQUEST_FILENAME}	Absolute filesystem path resolved from the web root for the requested URL
{URL}	URL path of the current request
{REQUEST_URI}	Same as {URL}

Action element attributes:

Attribute	Values	Default	Description
type	Rewrite / Redirect	—	Action to take when the rule matches
url	string	—	Target path or URL. Supports back-references {R:0}, {R:1}, and so on
appendQueryString	true / false	true	Append the original query string to the rewritten or redirected URL
redirectType	Permanent / Found / Temporary	Found	HTTP status code for Redirect actions (Permanent = 301, Found = 302, Temporary = 302)

Back-references from the match pattern are available in the action URL as `{R:0}` (full match) and `{R:1}`, `{R:2}`, … (capture groups).

6.3.2 HTTP Redirect

Redirect all requests from the site to a fixed destination URL:

```
<configuration>
  <system.webServer>
    <httpRedirect enabled="true"
                  destination="https://www.newsite.com"
                  exactDestination="false"
                  childOnly="false"
```

```
                    httpResponseStatus="Permanent" />
  </system.webServer>
</configuration>
```

httpRedirect attributes:

Attribute	Values	Default	Description
enabled	true / false	false	Activate the redirect
destination	URL string	—	Target redirection URL
exactDestination	true / false	false	Redirect every request to the exact destination URL without appending the original path
childOnly	true / false	false	Only redirect child paths, not the root itself
httpResponseStatus	Permanent / Found / Temporary	Permanent	Permanent = 301, Found = 302, Temporary = 302

6.3.3 Custom Error Pages

Map HTTP status codes to custom error files:

```
<configuration>
  <system.webServer>
    <httpErrors>
      <error statusCode="404" path="/error-pages/404.asp" responseMode="E
xecuteURL" />
      <error statusCode="500" path="/error-pages/500.html" responseMode="
File" />
    </httpErrors>
  </system.webServer>
</configuration>
```

error element attributes:

Attribute	Values	Description
statusCode	integer	HTTP status code to intercept
path	string	Relative URL or file path for the custom error page
responseMode	File / ExecuteURL	File serves a static file; ExecuteURL executes the path as an ASP page

Custom error mappings defined in `web.config` override the default error pages directory set via `default_error_pages_directory` in `axonasp.toml`.

6.4 Remarks

- web.config is read only from the web root directory. Files in subdirectories are not processed.
- The `httpRedirect` directive is evaluated before rewrite rules.
- URL rewrite rules are applied before ASP script execution and before default page resolution.
- Rules are evaluated in document order. `stopProcessing="true"` prevents further rule evaluation once a rule matches.
- The `{REQUEST_FILENAME}` input resolves to an absolute filesystem path rooted at the configured web root.
- Regex patterns use Go RE2 syntax, which does not support lookahead or backreferences.
- Paths in `httpErrors` are resolved relative to the web root.

6.5 Code Example

Route all non-file, non-directory requests through a single ASP front controller:

```
<configuration>
  <system.webServer>
    <rewrite>
      <rules>
        <rule name="Single Entry Point" stopProcessing="true">
          <match url="^(.+)$" ignoreCase="true" />
          <conditions logicalGrouping="MatchAll">
            <add input="{REQUEST_FILENAME}" matchType="IsFile" negate="tr
ue" />
            <add input="{REQUEST_FILENAME}" matchType="IsDirectory" negat
e="true" />
          </conditions>
          <action type="Rewrite" url="/index.asp" appendQueryString="true
" />
        </rule>
      </rules>
    </rewrite>
  </system.webServer>
</configuration>
```

7 MyInfo.xml

7.1 Overview

`MyInfo.xml` is a legacy XML configuration file originally introduced by Microsoft Personal Web Server (PWS). AxonASP provides compatibility support for this file through the `MSWC.MyInfo` object. When an ASP page creates an instance of `MSWC.MyInfo`, the server reads `MyInfo.xml` from the web root and exposes its XML element values as readable object properties.

7.2 File Location

Place `MyInfo.xml` at the root of the configured web root directory. The file is automatically blocked from direct HTTP access by the `blocked_files` setting in `axonasp.toml` — clients cannot download or view the file through the browser.

7.3 File Format

The file uses a flat XML structure with a `<MyInfo>` root element. Each child element name becomes an accessible property on the object:

```
<MyInfo>
  <PersonalName>Site Administrator</PersonalName>
  <PersonalAddress>123 Server Lane, Datacenter City</PersonalAddress>
  <PersonalPhone>+1 555-0100</PersonalPhone>
  <PersonalMail>admin@example.com</PersonalMail>
  <PersonalWords>Welcome to our site!</PersonalWords>
  <CompanyName>My Company</CompanyName>
  <URL1>http://www.example.com</URL1>
  <URLWords1>Company Website</URLWords1>
  <URL2>http://blog.example.com</URL2>
  <URLWords2>Company Blog</URLWords2>
</MyInfo>
```

7.4 Usage

Instantiate the object with `Server.CreateObject` using PROGID `MSWC.MyInfo`:

```
<%
Dim mi
Set mi = Server.CreateObject("MSWC.MyInfo")

Response.Write mi.PersonalName & "<br>"
Response.Write mi.CompanyName & "<br>"
Response.Write mi.URL(1) & "<br>"
Response.Write mi.URLWords(1) & "<br>"
%>
```

7.5 Properties

All XML element names in `MyInfo.xml` are exposed as properties on the object. Property names are case-insensitive.

Property	Type	Description
PersonalName	String	Administrator or site owner name
PersonalAddress	String	Physical or mailing address
PersonalPhone	String	Contact phone number
PersonalMail	String	Contact email address
PersonalWords	String	Welcome message or site tagline
CompanyName	String	Organization name
URL(n)	String	Indexed URL entry. Access as a method call with the numeric index: `mi.URL(1)`
URLWords(n)	String	Indexed URL label. Access as a method call with the numeric index: `mi.URLWords(1)`
(any element)	String	Any custom element added to `MyInfo.xml` is accessible as a property with the same name

7.6 Remarks

- Properties are read-only. Assigning a value to a property has no effect and does not modify the XML file.
- The object reads `MyInfo.xml` once at construction time. Changes to the file are reflected on the next `Server.CreateObject("MSWC.MyInfo")` call (or next page request).
- `URL` and `URLWords` entries use a numeric suffix in the XML (`URL1`, `URLWords1`, `URL2`, `URLWords2`). They are accessed in ASP code as method calls with a numeric argument: `mi.URL(1)`, `mi.URLWords(2)`.
- All other custom properties defined as XML child elements can be read directly as named properties.
- The file is resolved using `Server.MapPath("MyInfo.xml")`, which places it relative to the current web root.
- If the XML contains duplicate element names, only the last occurrence is returned.

7.7 Code Example

Display site information from `MyInfo.xml` on a contact page:

```
<%
Dim mi
Set mi = Server.CreateObject("MSWC.MyInfo")
%>
<p><strong>Site Administrator:</strong> <%= mi.PersonalName %></p>
<p><strong>Company:</strong> <%= mi.CompanyName %></p>
<p><strong>Contact:</strong> <a href="mailto:<%= mi.PersonalMail %>"><%=
mi.PersonalMail %></a></p>
<p><strong>Phone:</strong> <%= mi.PersonalPhone %></p>
<p><strong>Message:</strong> <%= mi.PersonalWords %></p>
<ul>
<%
Dim i
For i = 1 To 5
    Dim url, label
    url = mi.URL(i)
    label = mi.URLWords(i)
    If url <> "" Then
%>
    <li><a href="<%= url %>"><%= label %></a></li>
<%
    End If
Next
%>
</ul>
```

8 Script Caching

8.1 Overview

AxonASP uses a multi-tier bytecode caching system to eliminate redundant compilation of ASP scripts and dynamic code expressions. Once a script is compiled the first time, subsequent requests reuse the cached bytecode directly, bypassing the lexer and compiler entirely. Caching operates across four independent subsystems depending on the source of the compiled code.

8.2 Cache Tiers

8.2.1 1. Script Cache (axonvm/script_cache.go)

The primary cache stores compiled bytecode for ASP page files on disk. The file cache operates in two tiers:

- **Memory tier (Tier 1):** An LRU in-memory cache keyed by the source file path. Fastest lookup; no disk I/O.
- **Disk tier (Tier 2):** A binary cache file stored under `temp/cache`. File content is hashed with xxhash (64-bit). On cache miss the server compiles the script and saves the compiled payload to disk.

The disk format includes a `G3AXON` magic header, a binary version number, the source file modification timestamp, and the full compiled program payload. On load, the cached entry's modification time is verified against the source file. Stale entries are rejected and the script is recompiled.

A file watcher (`fsnotify`) monitors the web root for changes and automatically invalidates in-memory cache entries when a source file is modified. This means page edits are picked up immediately without restarting the server.

> **Note**: AxonASP's caching mechanism uses a file watcher to detect file modifications. This can occasionally result in a stale cache or slow updates, causing issues in development environments. To prevent this, you might want to disable the cache during ASP development by setting bytecode_caching_enabled=disabled.

Cache modes:

Mode	Memory tier	Disk tier
enabled	Yes	Yes
memory-only	Yes	No
disk-only	No	Yes
disabled	No	No

Configure the mode in `config/axonasp.toml`:

```
[global]
bytecode_caching_enabled = "enabled"
cache_max_size_mb = 128
```

8.2.2 2. Eval Cache (axonvm/eval_cache.go)

The Eval cache stores compiled bytecode for expressions passed to the VBScript `Eval()` built-in function. Expressions are keyed by a 64-bit FNV-1a hash derived from:

- The expression text
- The current `Option Compare` mode
- A hash of the active global symbol names
- A hash of the local procedure scope symbol names

Cache capacity is fixed at 512 entries (LRU eviction). An expression that produces the same hash as a cached entry is reused directly without recompilation. The cache is process-wide and shared across all concurrent requests.

8.2.3 3. Dynamic Execute Cache (axonvm/dynamic_exec_cache.go)

The dynamic execute cache stores compiled bytecode for code fragments passed to the VBScript `Execute()` and `ExecuteGlobal()` built-in functions. The cache key is a 64-bit FNV-1a hash derived from:

- The code fragment text
- The execution kind (Execute vs ExecuteGlobal)
- The current `Option Compare` and `Option Explicit` modes
- A hash of the active global symbol names and local scope (for Execute)
- A snapshot of active class version and class name (for Execute)

Cache capacity is fixed at 512 entries (LRU eviction). On a cache hit, the compiler symbol snapshot is also restored to keep class and global state consistent.

8.2.4 4. Execute File Cache (axonvm/execute_file_cache.go)

A dedicated in-memory-only cache with a capacity of 64 entries handles child pages invoked via `Server.Execute`. This cache uses the same `ScriptCache` subsystem as the primary script cache but operates in `memory-only` mode without disk persistence.

8.3 Configuration Reference

Setting	Section	Type	Default	Description
bytecode_caching_enabled	[global]	string	“enabled”	Cache mode: enabled / memory-only / disk-only / disabled
cache_max_size_mb	[global]	integer	128	Maximum size of the in-memory script cache in megabytes
clean_cache_on_startup	[global]	boolean	true	Clear compiled bytecode cache files at server startup

8.4 Remarks

- Caching is fully transparent. No code changes are needed in ASP scripts to benefit from it.
- The disk cache is stored in `temp/cache`. This directory is created automatically if it does not exist.
- On clean server startup with `clean_cache_on_startup = true`, all existing disk cache files are deleted and a fresh compilation pass occurs on the first request to each script.

- The Eval and dynamic execute caches are always active when the VM is running, regardless of the `bytecode_caching_enabled` setting. Their size is fixed in code and is not controlled by `cache_max_size_mb`.
- The file watcher invalidates only memory-tier entries. Disk cache files are invalidated by the modification time check on load.
- Include file dependencies are tracked in the disk cache. If an included `.asp` or `.inc` file changes, the cache entry for the parent page is also invalidated.
- The binary cache format includes a version number. When the engine is upgraded and the binary version changes, all existing disk cache files are automatically rejected and recompiled.

9 Locale Support and Language Formatting

9.1 Overview

AxonASP provides comprehensive locale support for date, time, number, and currency formatting. The server supports 43 language and region variants, enabling global ASP applications to render content in multiple languages and formats without manual conversion.

Locale settings control how the following functions format and parse data:

- **Date/Time Functions:** `FormatDateTime`, `DateValue`, `TimeValue`, `CDate`, `DateAdd`, `DateDiff`, `DatePart`, `Weekday`
- **Number Functions:** `FormatNumber`, `FormatPercent`, `FormatCurrency`
- **Day/Month Names:** `MonthName`, `WeekdayName`
- **Implicit Conversions:** Date and number values converted to strings

9.2 Working with Locales

9.2.1 Setting the Locale

The locale is controlled by the **Session.LCID** property. By default, it inherits from the server's configured default locale (`global.default_mslcid` in `axonasp.toml`).

In ASP Code:

```
' Set locale for Portuguese Brazil
Session.LCID = 1046

' Format a number using Portuguese locale
amount = 1234.56
formatted = FormatNumber(amount, 2)  ' Outputs: 1.234,56

' Format currency
currency = FormatCurrency(amount)    ' Outputs: R$ 1.234,56
```

Checking Current Locale:

```
currentLocale = Session.LCID
Response.Write "Current locale: " & currentLocale
```

9.2.2 Server Default Locale

The server default is configured in `axonasp.toml`:

```
[global]
default_mslcid = 1046  # Portuguese (Brazil)
```

This default applies to all new sessions. Individual requests can override it using `Session.LCID`.

9.3 Supported Locales and Variants

AxonASP supports the following 43 language and region locales:

9.3.1 English Variants

Locale	LCID	Region	Decimal	Thousands	Currency
en-US	1033	United States	.	,	$
en-GB	2057	United Kingdom	.	,	£
en-AU	3081	Australia	.	,	$
en-CA	4105	Canada	.	,	$
en-IN	16393	India	.	,	₹
en-IE	6153	Ireland	.	,	€
en-NZ	5129	New Zealand	.	,	$
en-ZA	7177	South Africa	.	,	R

9.3.2 Portuguese Variants

Locale	LCID	Region	Decimal	Thousands	Currency
pt-BR	1046	Brazil	,	.	R$
pt-PT	2070	Portugal	,	.	€

9.3.3 Spanish Variants

Locale	LCID	Region	Decimal	Thousands	Currency
es-ES	1034	Spain	,	.	€
es-MX	2058	Mexico	.	,	$
es-AR	11274	Argentina	,	.	$
es-CO	9226	Colombia	,	.	$
es-CL	13322	Chile	,	.	$

Locale	LCID	Region	Decimal	Thousands	Currency
es-PE	10250	Peru	.	,	S/

9.3.4 French Variants

Locale	LCID	Region	Decimal	Thousands	Currency
fr-FR	1036	France	,	(space)	€
fr-CA	3084	Canada	,	(space)	$
fr-BE	2060	Belgium	,	(space)	€
fr-CH	4108	Switzerland	.	’	CHF

9.3.5 German Variants

Locale	LCID	Region	Decimal	Thousands	Currency
de-DE	1031	Germany	,	.	€
de-AT	3079	Austria	,	.	€
de-CH	2055	Switzerland	.	’	CHF

9.3.6 Italian Variants

Locale	LCID	Region	Decimal	Thousands	Currency
it-IT	1040	Italy	,	.	€
it-CH	4108	Switzerland	.	’	CHF

9.3.7 Dutch Variants

Locale	LCID	Region	Decimal	Thousands	Currency
nl-NL	1043	Netherlands	,	.	€
nl-BE	2060	Belgium	,	.	€

9.3.8 Scandinavian Variants

Locale	LCID	Region	Decimal	Thousands	Currency
da-DK	1030	Denmark	,	.	kr
fi-FI	1035	Finland	,	(space)	€
nb-NO	1044	Norway	,	(space)	kr

9.3.9 Slavic Variants

Locale	LCID	Region	Decimal	Thousands	Currency
pl-PL	1045	Poland	,	(space)	zł
cs-CZ	1029	Czech Republic	,	(space)	Kč
bg-BG	1026	Bulgaria	,	(space)	лв
ru-RU	1049	Russia	,	(space)	₽
uk-UA	1058	Ukraine	,	(space)	₴

Locale	LCID	Region	Decimal	Thousands	Currency
sk-SK	1051	Slovakia	,	(space)	€
hr-HR	1050	Croatia	,	.	€

9.3.10 Asian Variants

Locale	LCID	Region	Decimal	Thousands	Currency
zh-CN	2052	China	.	,	¥
zh-TW	1028	Taiwan	.	,	NT$
zh-HK	3076	Hong Kong	.	,	HK$
ja-JP	1041	Japan	.	,	¥
ko-KR	1042	Korea	.	,	₩
th-TH	1054	Thailand	.	,	฿

9.3.11 Other Variants

Locale	LCID	Region	Decimal	Thousands	Currency
el-GR	1032	Greece	,	.	€
id-ID	1057	Indonesia	,	.	Rp
tr-TR	1055	Turkey	,	.	₺

9.4 Formatting Examples

9.4.1 Number Formatting

Numbers are formatted with locale-specific decimal and thousands separators:

ASP Code:

```
Dim amount, formatted

amount = 1234.567
Session.LCID = 1033  ' English US

formatted = FormatNumber(amount, 2)     ' Output: 1,234.57
formatted = FormatPercent(0.125, 1)     ' Output: 12.5%

' Switch to German
Session.LCID = 1031
formatted = FormatNumber(amount, 2)     ' Output: 1.234,57
formatted = FormatPercent(0.125, 1)     ' Output: 12,5%

' Switch to French
Session.LCID = 1036
formatted = FormatNumber(amount, 2)     ' Output: 1 234,57
formatted = FormatPercent(0.125, 1)     ' Output: 12,5%
```

9.4.2 Currency Formatting

Currency symbols and separators are locale-aware:

```
Dim price

price = 1234.50
Session.LCID = 1033  ' English US
Response.Write FormatCurrency(price)    ' Output: $1,234.50

Session.LCID = 1046  ' Portuguese Brazil
Response.Write FormatCurrency(price)    ' Output: R$ 1.234,50

Session.LCID = 1036  ' French France
Response.Write FormatCurrency(price)    ' Output: € 1 234,50
```

9.4.3 Date Formatting

Dates are formatted according to locale conventions:

```
Dim eventDate

eventDate = CDate("2026-04-09")
Session.LCID = 1033  ' English US
Response.Write FormatDateTime(eventDate, 1)  ' Output: Thursday, April 9,
 2026

Session.LCID = 1046  ' Portuguese Brazil
Response.Write FormatDateTime(eventDate, 1)  ' Output: quinta-feira, 9 de
 abril de 2026

Session.LCID = 1036  ' French France
Response.Write FormatDateTime(eventDate, 1)  ' Output: jeudi 9 avril 2026
```

9.4.4 Month and Day Names

Month and weekday names adapt to the session locale:

```
Session.LCID = 1046  ' Portuguese Brazil

Response.Write MonthName(4)         ' Output: abril
Response.Write MonthName(4, True)   ' Output: abr
Response.Write WeekdayName(1)       ' Output: domingo
Response.Write WeekdayName(1, True) ' Output: dom
```

9.4.5 Date Parsing

Date parsing respects locale conventions when parsing string dates:

```
Dim dateStr

Session.LCID = 1046  ' Portuguese Brazil
dateStr = "09/04/2026"  ' Day/Month/Year format
Response.Write CDate(dateStr)  ' Parses as: April 9, 2026

Session.LCID = 1033  ' English US
dateStr = "04/09/2026"  ' Month/Day/Year format
Response.Write CDate(dateStr)  ' Parses as: April 9, 2026
```

9.5 Affected Functions

The following VBScript functions respect the current `Session.LCID`:

9.5.1 Date/Time Functions

- `FormatDateTime(expression, format)` - All format types (1-4)
- `DateValue(expression)` - Parses dates in locale format
- `TimeValue(expression)` - Parses times
- `CDate(expression)` - Parses dates and times
- `DateAdd(interval, number, date)` - Returns date/time values
- `DateDiff(interval, date1, date2)` - Returns intervals
- `DatePart(interval, date)` - Extracts date parts
- `Day(date)`, `Month(date)`, `Year(date)` - Date component extraction
- `Weekday(date)` - Returns day of week (1 = Sunday, 7 = Saturday)
- `MonthName(number, abbrev)` - Locale-aware month names
- `WeekdayName(index, abbrev)` - Locale-aware weekday names

9.5.2 Number Functions

- `FormatNumber(expression, numDigits, useGrouping)` - Uses locale separators
- `FormatPercent(expression, numDigits, useGrouping)` - Uses locale separators
- `FormatCurrency(expression, numDigits, useGrouping)` - Uses locale symbol and separators

9.5.3 String Conversion

- Implicit VTDate to string conversion respects locale date/time format
- Implicit numeric to string conversion uses locale separators

9.6 Performance Notes

Locale resolution is optimized for performance:

- Locale profiles are precomputed at server startup

- Language matching uses efficient BCP 47 tag matching
- Formatting functions cache locale profiles in the VM context
- No reflection or dynamic dispatch is used in formatting paths

9.7 Configuration

The server default locale is configured in `axonasp.toml`:

```
[global]
default_mslcid = 1046        # Portuguese (Brazil)
default_timezone = "America/Sao_Paulo"
```

The `default_timezone` setting affects date/time implicit conversions and parsing. Both UTC and named timezones (IANA timezone database) are supported.

9.8 Troubleshooting

9.8.1 Incorrect Date Parsing

If dates parse incorrectly, verify the session locale matches the date format:

```
' Bad: locale mismatch
Session.LCID = 1033  ' US (MM/DD/YYYY)
Response.Write CDate("25/12/2026")  ' Fails - US format expects MM/DD

' Good: locale matches format
Session.LCID = 1046  ' Brazil (DD/MM/YYYY)
Response.Write CDate("25/12/2026")  ' Works correctly
```

9.8.2 Missing Locale

If a specific locale code is not in the supported list, the system falls back to the closest matching locale (e.g., an unmapped Spanish variant falls back to es-ES).

9.8.3 Currency Symbol Not Appearing

Ensure the locale is set before calling `FormatCurrency`:

```
Session.LCID = 1046  ' Must be set before formatting
Response.Write FormatCurrency(100)
```

10 FastCGI Setup

10.1 Overview

AxonASP provides a FastCGI application server (`axonasp-fastcgi.exe`) that integrates directly with Nginx, Apache, or IIS using the FastCGI protocol. In this mode AxonASP acts as a backend process and the front-end web server handles all HTTP connections, static content, and TLS termination.

AxonASP FastCGI fully supports **multi-host deployments** with different document roots, similar to PHP-FPM. This allows a single FastCGI server process to serve content from multiple virtual hosts, each with its own document root directory.

10.2 Prerequisites

- `axonasp-fastcgi.exe` running and reachable on the configured port (default: 9000)
- Front-end web server with FastCGI support installed (nginx, Apache, IIS)

The FastCGI port is configured in `config/axonasp.toml`:

```toml
[fastcgi]
server_port = 9000
```

The port can also be a socket path on Linux and macOS:

```toml
server_port = "unix:/tmp/axonasp.sock"
```

Start AxonASP FastCGI:

```
.\axonasp-fastcgi.exe
```

10.3 DOCUMENT_ROOT and SCRIPT_NAME Parameters

AxonASP FastCGI reads the following FastCGI CGI variables to support multi-host deployments:

- **`DOCUMENT_ROOT`**: The directory where the virtual host's files are located. When provided by the reverse proxy, files are resolved from this directory instead of the configured `RootDir`.
- **`SCRIPT_NAME`**: The virtual path to the requested ASP script (e.g., `/default.asp`). When provided, this takes priority over the URL path for path resolution.

If `DOCUMENT_ROOT` is not provided, the server falls back to the `RootDir` configured in the TOML file, ensuring backward compatibility with existing single-host setups.

10.4 Nginx with FastCGI

10.4.1 Single Virtual Host

```
upstream axonasp_fcgi {
    server localhost:9000 max_fails=3 fail_timeout=30s;
}

server {
    listen 443 ssl http2;
    server_name myapp.example.com;

    ssl_certificate /etc/ssl/certs/myapp.crt;
```

```
    ssl_certificate_key /etc/ssl/private/myapp.key;
    ssl_protocols TLSv1.2 TLSv1.3;
    ssl_ciphers HIGH:!aNULL:!MD5;

    root /var/www/myapp;
    index default.asp index.asp;

    location ~ \.asp$ {
        fastcgi_pass axonasp_fcgi;
        # CRITICAL: Pass DOCUMENT_ROOT for multi-host support
        fastcgi_param DOCUMENT_ROOT $document_root;
        fastcgi_param SCRIPT_NAME $fastcgi_script_name;
        fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
        fastcgi_param REQUEST_METHOD $request_method;
        fastcgi_param QUERY_STRING $query_string;
        fastcgi_param SERVER_NAME $server_name;
        fastcgi_param SERVER_PORT $server_port;
        fastcgi_param REQUEST_URI $request_uri;
        fastcgi_param DOCUMENT_URI $document_uri;
        fastcgi_param HTTPS $https;
        include fastcgi_params;
    }

    location / {
        try_files $uri $uri/ =404;
    }
}
```

10.4.2 Multiple Virtual Hosts (Single FastCGI Process)

The primary advantage of AxonASP FastCGI is supporting multiple virtual hosts with different document roots from a **single FastCGI server process**. This architecture is identical to PHP-FPM:

```
upstream axonasp_fcgi {
    server localhost:9000 max_fails=3 fail_timeout=30s;
}

# Virtual Host #1
server {
    listen 443 ssl http2;
    server_name site1.example.com;
    root /var/www/site1;

    ssl_certificate /etc/ssl/certs/site1.crt;
    ssl_certificate_key /etc/ssl/private/site1.key;

    location ~ \.asp$ {
        fastcgi_pass axonasp_fcgi;
```

```
        fastcgi_param DOCUMENT_ROOT $document_root;
        fastcgi_param SCRIPT_NAME $fastcgi_script_name;
        fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
        include fastcgi_params;
    }
}

# Virtual Host #2
server {
    listen 443 ssl http2;
    server_name site2.example.com;
    root /var/www/site2;

    ssl_certificate /etc/ssl/certs/site2.crt;
    ssl_certificate_key /etc/ssl/private/site2.key;

    location ~ \.asp$ {
        fastcgi_pass axonasp_fcgi;
        fastcgi_param DOCUMENT_ROOT $document_root;
        fastcgi_param SCRIPT_NAME $fastcgi_script_name;
        fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
        include fastcgi_params;
    }
}
```

How it works: 1. Nginx receives request for `site1.example.com/index.asp` 2. Nginx sets `$document_root = /var/www/site1` and passes it to FastCGI 3. AxonASP FastCGI receives `DOCUMENT_ROOT=/var/www/site1` parameter 4. AxonASP loads `/var/www/site1/index.asp` (not from the configured RootDir)

This enables true multi-tenant ASP hosting from a single FastCGI process.

10.5 Key FastCGI Parameters

Parameter	Description	Required
`DOCUMENT_ROOT`	Directory containing the virtual host's files	Yes (for multi-host)
`SCRIPT_NAME`	Virtual path to the ASP script	Yes (for proper path resolution)
`SCRIPT_FILENAME`	Absolute file path (informational, not used by AxonASP)	No
`REQUEST_METHOD`	HTTP method (GET, POST, etc.)	Yes
`QUERY_STRING`	URL query parameters	Yes
`SERVER_NAME`	Virtual host name	Yes

Parameter	Description	Required
SERVER_PORT	HTTP/HTTPS port	Yes
HTTPS	“on” if HTTPS, “off” if HTTP	Yes

10.6 Apache with FastCGI

Apache with mod_fcgid or mod_proxy_fcgi automatically passes FastCGI CGI environment variables, including DOCUMENT_ROOT and SCRIPT_NAME, to the FastCGI application.

10.6.1 mod_fcgid Configuration

```
<VirtualHost *:443>
    ServerName myapp.example.com
    DocumentRoot "/var/www/myapp"

    SSLEngine on
    SSLCertificateFile /etc/ssl/certs/myapp.crt
    SSLCertificateKeyFile /etc/ssl/private/myapp.key

    <IfModule mod_fcgid.c>
        FcgidConnectTimeout 30
        FcgidIdleTimeout 300
        FcgidMaxRequestLen 1073741824
    </IfModule>

    <FilesMatch "\.asp$">
        SetHandler fcgid-script
    </FilesMatch>

    FcgidWrapper "unix:/var/run/axonasp.sock" .asp
    # or FcgidWrapper "127.0.0.1:9000" .asp
</VirtualHost>
```

10.6.2 mod_proxy_fcgi Configuration

```
<VirtualHost *:443>
    ServerName myapp.example.com
    DocumentRoot "/var/www/myapp"

    SSLEngine on
    SSLCertificateFile /etc/ssl/certs/myapp.crt
    SSLCertificateKeyFile /etc/ssl/private/myapp.key

    <FilesMatch "\.asp$">
        SetHandler "proxy:unix:/var/run/axonasp.sock|fcgi://localhost/"
        # or: SetHandler "proxy:fcgi://127.0.0.1:9000"
    </FilesMatch>
</VirtualHost>
```

Enable required modules:

```
a2enmod fcgid ssl
# or for proxy_fcgi:
a2enmod proxy proxy_fcgi ssl
systemctl restart apache2
```

Apache automatically sets `DOCUMENT_ROOT` to the `DocumentRoot` directive value and `SCRIPT_NAME` to the requested script path, enabling seamless multi-host support.

10.7 IIS with FastCGI

Configure IIS to use AxonASP as a FastCGI handler for `.asp` files:

```
# Run as Administrator
Import-Module WebAdministration

# Add FastCGI application
Add-WebConfiguration -PSPath "IIS:\" `
    -Filter "system.webServer/fastCgi" `
    -Value @{
        fullPath = "C:\axonasp\axonasp-fastcgi.exe";
        arguments = "";
        instanceMaxRequests = 10000;
        maxInstances = 4;
        requestTimeout = 300;
    }

# Map .asp extension to the FastCGI handler
Add-WebConfigurationProperty `
    -PSPath "IIS:\Sites\Default Web Site" `
    -Filter "system.webServer/handlers" `
    -Name "." `
    -Value @{
        name = "AxonASP-FastCGI";
        path = "*.asp";
        verb = "*";
        modules = "FastCgiModule";
        scriptProcessor = "C:\axonasp\axonasp-fastcgi.exe";
        resourceType = "File";
        requireAccess = "Read";
    }

iisreset
```

10.8 Unix Socket Mode

On Linux and macOS you can use a Unix domain socket instead of a TCP port for lower overhead:

```
[fastcgi]
server_port = "unix:/tmp/axonasp.sock"
```

Configure Nginx to connect via the socket:

```
upstream axonasp_fcgi {
    server unix:/tmp/axonasp.sock;
}
```

10.9 Remarks

- The FastCGI server supports the same ASP libraries and functions as the HTTP server. Feature parity is maintained between all runtime modes.
- `global.asa` is loaded from the configured web root on startup, identical to the HTTP server, you should set the `server.web_root` key to the parent directory of `global.asa` for it to be loaded properly.
- The FastCGI server does not serve static files directly. The front-end web server handles static content.
- Increase `instanceMaxRequests` or `maxInstances` in IIS for high-traffic deployments.
- For Nginx, use `fastcgi_read_timeout` to match the AxonASP script timeout configured in `axonasp.toml`.

11 Reverse Proxy Configuration

11.1 Overview

The recommended way to deploy AxonASP in production is behind a reverse proxy. The proxy handles TLS termination, rate limiting, static asset caching, and load balancing, while AxonASP focuses on executing ASP scripts. AxonASP HTTP server listens on port 8801 by default.

11.2 Prerequisites

- AxonASP HTTP server (`axonasp-http.exe`) running and reachable on localhost
- A reverse proxy (Nginx, Apache, Caddy, or IIS) installed on the same machine or network

Start AxonASP:

```
.\axonasp-http.exe
```

11.3 Why Not Expose AxonASP Directly

Do not expose `axonasp-http.exe` directly to public internet traffic:

- No centralized TLS/SSL termination

- No rate-limiting or DDoS protection
- No centralized request logging
- Increased attack surface

Use a reverse proxy as the public-facing entry point.

11.4 Nginx (Proxy Mode)

```
upstream axonasp_backend {
    server localhost:8801 max_fails=3 fail_timeout=30s;
}

server {
    listen 80;
    server_name myapp.example.com;
    return 301 https://$server_name$request_uri;
}

server {
    listen 443 ssl http2;
    server_name myapp.example.com;

    ssl_certificate /etc/ssl/certs/myapp.crt;
    ssl_certificate_key /etc/ssl/private/myapp.key;
    ssl_protocols TLSv1.2 TLSv1.3;
    ssl_ciphers HIGH:!aNULL:!MD5;

    location / {
        proxy_pass http://axonasp_backend;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
        proxy_buffering off;
        proxy_request_buffering off;
    }
}
```

11.5 Apache (Proxy Mode)

```
<VirtualHost *:80>
    ServerName myapp.example.com
    RewriteEngine On
    RewriteCond %{HTTPS} off
    RewriteRule ^(.*)$ https://%{HTTP_HOST}%{REQUEST_URI} [L,R=301]
</VirtualHost>

<VirtualHost *:443>
    ServerName myapp.example.com
```

```
    SSLEngine on
    SSLCertificateFile /etc/ssl/certs/myapp.crt
    SSLCertificateKeyFile /etc/ssl/private/myapp.key
    SSLProtocol TLSv1.2 TLSv1.3

    ProxyRequests Off
    ProxyPreserveHost On

    <Location />
        ProxyPass http://localhost:8801/
        ProxyPassReverse http://localhost:8801/
        RequestHeader set X-Real-IP "%{REMOTE_ADDR}s"
        RequestHeader set X-Forwarded-For "%{HTTP:X-Forwarded-For}s, %{RE
MOTE_ADDR}s"
        RequestHeader set X-Forwarded-Proto "%{REQUEST_SCHEME}s"
    </Location>
</VirtualHost>
```

Enable required modules:

```
a2enmod proxy proxy_http ssl rewrite headers
systemctl restart apache2
```

11.6 Caddy (Proxy Mode)

Caddy handles HTTPS certificates automatically through Let's Encrypt:

```
myapp.example.com {
    reverse_proxy localhost:8801 {
        header_up X-Forwarded-Proto {scheme}
        header_up X-Forwarded-For {remote_host}
        header_up X-Real-IP {remote_host}
    }
}
```

Start Caddy:

```
caddy run
```

11.7 IIS (Proxy Mode)

Configure IIS Application Request Routing to forward requests to AxonASP:

```
<!-- web.config on the IIS site -->
<configuration>
  <system.webServer>
    <rewrite>
      <rules>
        <rule name="Proxy to AxonASP" stopProcessing="true">
          <match url="(.*)" />
```

```
          <action type="Rewrite" url="http://localhost:8801/{R:1}" />
        </rule>
      </rules>
    </rewrite>
  </system.webServer>
</configuration>
```

Requires IIS Application Request Routing (ARR) and URL Rewrite modules.

11.8 Load Balancing Multiple Instances

For high-traffic deployments, run multiple AxonASP instances on different ports and balance across them:

```
upstream axonasp_cluster {
    server localhost:8801 weight=1;
    server localhost:8802 weight=1;
    server localhost:8803 weight=1;
}

server {
    listen 443 ssl http2;
    server_name myapp.example.com;

    location / {
        proxy_pass http://axonasp_cluster;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
    }
}
```

Start each instance on a different port using environment variables:

```
# Terminal 1
$env:SERVER_PORT = "8801"; .\axonasp-http.exe

# Terminal 2
$env:SERVER_PORT = "8802"; .\axonasp-http.exe

# Terminal 3
$env:SERVER_PORT = "8803"; .\axonasp-http.exe
```

11.9 Remarks

- AxonASP does not handle TLS/SSL natively. Terminate SSL at the reverse proxy and forward plain HTTP to AxonASP.

- Set `proxy_buffering off` in Nginx when using `Response.Flush` or streaming responses in ASP scripts.
- The port can also be changed in `config/axonasp.toml` under `[server] server_port`.
- All proxy configurations above assume AxonASP runs on the same machine as the proxy on localhost. Adjust the upstream address for remote deployments.

12 Running AxonASP as a Linux Service

12.1 Overview

This page describes how to deploy the AxonASP HTTP or FastCGI server as a persistent background service on Linux using `systemd`. Running AxonASP as a systemd service ensures it starts automatically on boot, restarts after crashes, and integrates with standard Linux service management tools.

12.2 Prerequisites

- Compiled `axonasp-http` or `axonasp-fastcgi` binary (built with `build.sh`)
- A Linux system with `systemd` (Ubuntu 18+, Debian 10+, CentOS 7+, and similar)
- A dedicated service account (recommended)

12.3 Building for Linux

Use `build.sh` in the project root:

```
./build.sh
```

The script builds the following binaries for the current platform: - `axonasp-http` — HTTP web server - `axonasp-fastcgi` — FastCGI application server - `axonasp-cli` — Command-line interpreter - `axonasp-mcp` — MCP server - `axonasp-service` — service wrapper helper

For cross-compilation to a specific platform/architecture:

```
./build.sh --platform linux --arch amd64
./build.sh --platform linux --arch arm64
```

12.4 Creating the Service User

```
sudo useradd --system --no-create-home --shell /usr/sbin/nologin axonasp
```

12.5 Deploying Files

Copy the binary, configuration, and web root to the target directory:

```
sudo mkdir -p /opt/axonasp
sudo cp axonasp-http /opt/axonasp/
sudo cp -r config /opt/axonasp/
```

```
sudo cp -r www /opt/axonasp/
sudo chown -R axonasp:axonasp /opt/axonasp
```

12.6 Creating the systemd Unit File

Create a unit file at `/etc/systemd/system/axonasp.service`:

```
[Unit]
Description=AxonASP HTTP Server
After=network.target
Wants=network-online.target

[Service]
Type=simple
User=axonasp
Group=axonasp
WorkingDirectory=/opt/axonasp
ExecStart=/opt/axonasp/axonasp-http
Restart=on-failure
RestartSec=5
StandardOutput=journal
StandardError=journal
SyslogIdentifier=axonasp

# Resource limits
LimitNOFILE=65536
LimitNPROC=4096

# Security hardening
NoNewPrivileges=true
PrivateTmp=true

[Install]
WantedBy=multi-user.target
```

For the **FastCGI server**, replace `ExecStart` with:

```
ExecStart=/opt/axonasp/axonasp-fastcgi
```

12.7 Enabling and Starting the Service

```
# Reload systemd to pick up the new unit file
sudo systemctl daemon-reload

# Enable the service to start on boot
sudo systemctl enable axonasp

# Start the service now
sudo systemctl start axonasp
```

12.8 Managing the Service

```
# Check status
sudo systemctl status axonasp

# Stop the service
sudo systemctl stop axonasp

# Restart the service
sudo systemctl restart axonasp

# View logs
sudo journalctl -u axonasp -f

# View last 100 log lines
sudo journalctl -u axonasp -n 100 --no-pager
```

12.9 Configuring the Web Root and Port

Edit `/opt/axonasp/config/axonasp.toml` to set the web root and port:

```
[server]
port = 8801
web_root = "/opt/axonasp/www"
```

These values can also be passed via environment variables in the unit file:

```
[Service]
Environment="WEB_ROOT=/opt/axonasp/www"
Environment="SERVER_PORT=8801"
```

12.10 Nginx Reverse Proxy Integration

Run AxonASP behind Nginx for TLS and public-facing traffic:

```
server {
    listen 443 ssl http2;
    server_name myapp.example.com;

    ssl_certificate     /etc/ssl/certs/myapp.crt;
    ssl_certificate_key /etc/ssl/private/myapp.key;

    location / {
        proxy_pass         http://127.0.0.1:8801;
        proxy_set_header   Host $host;
        proxy_set_header   X-Real-IP $remote_addr;
        proxy_set_header   X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header   X-Forwarded-Proto $scheme;
    }
}
```

12.11 Remarks

- Use `sudo journalctl -u axonasp` for real-time log monitoring instead of separate log files.
- The `PrivateTmp=true` directive gives the service its own isolated `/tmp` directory, preventing conflicts with other processes.
- If AxonASP writes session files or cache to `./temp/`, ensure that directory is owned by the `axonasp` user and writable.
- To update the binary, stop the service, replace the file, and restart: `sudo systemctl restart axonasp`.
- For high availability, multiple AxonASP instances can be run on different ports and load-balanced by the front-end proxy.

13 Installing AxonASP from Linux Packages

13.1 Overview

AxonASP publishes pre-built binary packages for Linux on every tagged release. These packages install all server binaries, the default configuration, the web root, and the required directory structure under `/opt/axonasp`. Installation is handled by the standard package manager of your distribution.

Packages are available in three formats:

- **.deb** — Debian, Ubuntu, and derivatives
- **.rpm** — Red Hat, Fedora, CentOS, Rocky Linux, and derivatives
- **.apk** — Alpine Linux

13.2 Prerequisites

- A 64-bit (amd64) Linux system
- Root or `sudo` access
- Internet access to reach the GitHub Releases page, or a locally downloaded package file

---

13.3 Downloading a Package

All packages are published to the GitHub Releases page. The download URL follows this pattern:

```
https://github.com/guimaraeslucas/axonasp/releases/download/<version>/<file>.<extension>
```

Replace `<version>` with the release tag (for example, `v2.0.1`), `<file>` with the package filename, and `<extension>` with `deb`, `rpm`, or `apk`.

Debian/Ubuntu example:

```
wget https://github.com/guimaraeslucas/axonasp/releases/download/v2.0.1/axonasp_2.0.1_linux_amd64.deb
```

Red Hat/Fedora/Rocky example:

```
wget https://github.com/guimaraeslucas/axonasp/releases/download/v2.0.1/axonasp-2.0.1-1.x86_64.rpm
```

Alpine example:

```
wget https://github.com/guimaraeslucas/axonasp/releases/download/v2.0.1/axonasp_2.0.1_x86_64.apk
```

You can check the actual filenames for a given release on the releases page at `https://github.com/guimaraeslucas/axonasp/releases`.

13.4 Installing the Package

13.4.1 Debian and Ubuntu

```
sudo dpkg -i axonasp_2.0.1_linux_amd64.deb
```

Or, to automatically resolve any missing recommendations:

```
sudo apt-get install -f ./axonasp_2.0.1_linux_amd64.deb
```

13.4.2 Red Hat, Fedora, CentOS, Rocky Linux

```
sudo rpm -i axonasp-2.0.1-1.x86_64.rpm
```

Or using `dnf`:

```
sudo dnf localinstall axonasp-2.0.1-1.x86_64.rpm
```

13.4.3 Alpine Linux

```
sudo apk add --allow-untrusted axonasp_2.0.1_x86_64.apk
```

13.5 What Gets Installed

After a successful installation, the following layout is created:

Path	Description
`/opt/axonasp/`	Main installation directory

Path	Description
/opt/axonasp/axonasp-http	HTTP web server binary
/opt/axonasp/axonasp-fastcgi	FastCGI application server binary
/opt/axonasp/axonasp-cli	Command-line interpreter binary
/opt/axonasp/axonasp-mcp	MCP server binary
/opt/axonasp/axonasp-testsuite	Automated test suite runner binary
/opt/axonasp/axonasp-service	Service wrapper helper binary
/opt/axonasp/config/axonasp.toml	Default configuration file
/opt/axonasp/www/	Default web root and documentation
/opt/axonasp/mcp/	MCP server resource files
/opt/axonasp/global.asa	Default global.asa application lifecycle file
/opt/axonasp/temp/cache/	Bytecode script cache directory
/opt/axonasp/temp/session/	Session storage directory

13.6 Symbolic Links

The package creates symbolic links so that the server binaries are accessible from any terminal without specifying the full path.

Symlink	Target
/usr/bin/axonasp-http	/opt/axonasp/axonasp-http
/usr/bin/axonasp-cli	/opt/axonasp/axonasp-cli
/usr/bin/axonasp-fastcgi	/opt/axonasp/axonasp-fastcgi
/usr/bin/axonasp-mcp	/opt/axonasp/axonasp-mcp
/usr/bin/axonasp-testsuite	/opt/axonasp/axonasp-testsuite
/etc/axonasp/config/axonasp.toml	/opt/axonasp/config/axonasp.toml
/var/cache/axonasp/	/opt/axonasp/temp/cache/
/var/lib/axonasp/session/	/opt/axonasp/temp/session/

Because the binaries are linked under /usr/bin, you can start the server from any working directory:

```
axonasp-http
```

The configuration symlink at /etc/axonasp/config/axonasp.toml provides a standard location for system administrators to locate and edit the configuration file without navigating to /opt/axonasp.

13.7 Starting the Server After Installation

After the package installs, start the HTTP server directly:

```
axonasp-http
```

To run it as a persistent background service, follow the instructions in the **Running AxonASP as a Linux Service** page of this manual. The package includes `install-service.sh` and `uninstall-service.sh` scripts under `/opt/axonasp/` that automate systemd unit file creation.

```
cd /opt/axonasp
sudo bash install-service.sh
sudo systemctl start axonasp
sudo systemctl enable axonasp
```

13.8 Remarks

- All packages target the **amd64 (x86-64)** architecture. ARM builds are not included in the pre-built packages; compile from source using `build.sh` for other architectures.
- The `temp/` subdirectories are created as empty directories by the package. The server writes bytecode caches and session data there at runtime.
- The configuration file at `/opt/axonasp/config/axonasp.toml` contains default values for all settings. Edit it before starting the server to configure the web root path, port, session behavior, and other options.
- The `www/tests/` directory is excluded from the distributed package. It is only present in the source repository.
- Package builds are triggered automatically on every push to the `main` branch and on every version tag. Non-tagged builds use a commit-count-based version number (`2.0.<commit-count>.<short-hash>`). Tagged releases use the clean tag version (for example, `2.0.1`).

14 AxonASP Service Wrapper

14.1 Overview

The AxonASP service wrapper starts and supervises either axonasp-http or axonasp-fastcgi as an operating system service.

This wrapper is intended to simplify service deployment, especially on Windows and small installations.

For advanced production environments, use the platform-native service strategy documented in the runtime manual pages, especially on Unix systems.

14.2 Binary Name

The build process generates:

- Windows: axonasp-service.exe
- Linux and macOS: axonasp-service

14.3 How It Works

- Reads service settings from the [service] section in config/axonasp.toml using the shared axonconfig loader.
- Resolves relative executable paths from the wrapper executable directory, not from the current working directory.
- This behavior is important on Windows, where services often start under system directories.
- Starts the configured child executable and monitors process lifetime.

14.4 Install and Manage the Service

14.5 Windows

Run commands from the deployment folder where axonasp-service.exe, config, and target binaries exist.

```
.\axonasp-service.exe install
.\axonasp-service.exe start
.\axonasp-service.exe stop
.\axonasp-service.exe uninstall
```

14.6 Linux or macOS

Use equivalent commands:

```
./axonasp-service install
./axonasp-service start
./axonasp-service stop
./axonasp-service uninstall
```

Notes:

- Service registration behavior depends on the host service manager.
- For complex Unix deployments, prefer dedicated systemd or launchd units and reverse-proxy orchestration described in other runtime pages.

14.7 Automated Installation and Uninstallation Scripts

Helper scripts are provided to simplify service installation and uninstallation with automatic status verification:

14.7.1 Windows

Use the PowerShell scripts from the project root:

Installation:

```
.\install-service.ps1
```

This script: - Checks for Administrator privileges. - Installs the service using axonasp-service.exe. - Starts the service automatically. - Verifies service status and reports success or failure.

Uninstallation:

```
.\uninstall-service.ps1
```

This script: - Checks for Administrator privileges. - Stops the running service (if active). - Uninstalls the service. - Verifies service removal from the system.

14.7.2 Linux and Unix Systems

Use the Bash scripts from the project root. These scripts require root privileges and assume systemd is available:

Installation:

```
sudo ./install-service.sh
```

This script: - Verifies root privileges. - Confirms systemd availability. - Installs the service using axonasp-service. - Starts the service automatically. - Verifies service status using systemctl.

Uninstallation:

```
sudo ./uninstall-service.sh
```

This script: - Verifies root privileges. - Confirms systemd availability. - Stops the running service (if active). - Uninstalls the service. - Verifies service removal from the system.

14.8 Service Configuration Flags in axonasp.toml

Configure the [service] section:

```
[service]
service_name = "AxonASPServer"
service_display_name = "G3pix AxonASP Server"
service_description = "AxonASP Service running AxonASP Server. This is a
wrapper used by axonasp-http or axonasp-fastcgi."
service_executable_path = "./axonasp-http"
service_environment_variables = []
```

14.8.1 service_name

Internal service identifier used by the operating system service manager.

14.8.2 service_display_name

Human-readable name shown in service management tools.

14.8.3 service_description

Service description displayed by the service manager.

14.8.4 service_executable_path

Path to child executable started by the wrapper.

- You can point to axonasp-http or axonasp-fastcgi.
- Relative paths are resolved from the wrapper executable directory.
- On Windows, .exe is automatically appended when no extension is provided.

14.8.5 service_environment_variables

List of environment variable entries in KEY=VALUE format applied to the child process.

Example:

```
service_environment_variables = ["SERVER_SERVER_PORT=9901"]
```

14.9 Deployment Recommendations

- Keep axonasp-service binary, config folder, and selected child executable in the same deployment root.
- Validate service_executable_path after each deployment.
- Keep this wrapper for straightforward local and Windows service setups.
- For advanced Unix operations, use native service manager definitions and process policies from the runtime Linux service guidance.

15 MCP Server and VS Code Integration

15.1 Overview

AxonASP includes an MCP (Model Context Protocol) server (`axonasp-mcp.exe`) that exposes the AxonASP documentation and tooling to AI assistants and IDEs that support MCP. The MCP server enables context-aware code generation, documentation lookup, and script analysis directly from your editor.

15.2 Starting the MCP Server

```
.\axonasp-mcp.exe
```

The MCP server operates in `stdio` mode by default, reading commands from standard input and writing responses to standard output. It can also be configured to run in SSE (Server-Sent Events) mode via `config/axonasp.toml`.

Configure the MCP mode in `config/axonasp.toml`:

```toml
[mcp]
mode = "stdio"
```

15.3 Configuring VS Code

To connect VS Code to the AxonASP MCP server, create a `.vscode` directory in the root of your project and add a `mcp.json` file with the following content:

File: `.vscode/mcp.json`

```json
{
    "servers": {
        "AxonASP MCP": {
            "type": "stdio",
            "command": ".\\axonasp-mcp.exe",
            "args": []
        }
    },
    "inputs": []
}
```

Steps:

1. Open your project root in VS Code.
2. Create a `.vscode` folder if it does not already exist.
3. Inside `.vscode`, create a file named `mcp.json`.
4. Paste the JSON configuration above into the file.
5. Ensure `axonasp-mcp.exe` is present in the project root directory.
6. Reload VS Code or activate the MCP extension to connect.

Once connected, AI tools in VS Code (such as GitHub Copilot) can query the AxonASP MCP server for documentation, object references, and code examples.

15.4 SSE Mode

The MCP server also supports SSE mode, which exposes an HTTP endpoint for browser-based or remote MCP clients.

```
[mcp]
mode = "sse"
sse_port = 8900
```

Start the server in SSE mode:

```
.\axonasp-mcp.exe
```

Connect remote clients to `http://localhost:8900`.

15.5 Remarks

- In `stdio` mode, the MCP server does not open a network port. VS Code launches the process directly and communicates through its stdin/stdout pipes.
- The MCP server shares the same configuration file (`config/axonasp.toml`) as the HTTP and FastCGI servers.
- Only one MCP transport mode (`stdio` or `sse`) is active at a time.
- The `.vscode/mcp.json` file is local to your workspace. Add it to `.gitignore` if you do not want to share the MCP configuration with other repository contributors.
- Maintainer note: The MCP server exposes two core tools: `search_axonasp_docs` for fuzzy documentation lookup and `get_asp_coding_style` for returning the full Classic ASP/VBScript formatting guide from `mcp/aspcodingstyle.md`. AI agents should call `get_asp_coding_style` whenever code-style or formatting rules are needed before generating or refactoring ASP code.

16 System Pages and Error Pages

16.1 Overview

AxonASP reserves two directories for internal server assets: `www/axonasp-pages/` contains styling, images, and the directory-listing template, while `www/error-pages/` contains custom HTTP error pages. Both directories are blocked from direct browser access by default.

16.2 axonasp-pages/

Path: `www/axonasp-pages/`

This directory holds static assets used internally by AxonASP server features. It is not intended for application code.

Contents:

Path	Description
`axonasp-pages/css/`	AxonASP built-in stylesheet used by the manual and system pages.
`axonasp-pages/images/`	Logo and icon assets for AxonASP system pages.
`axonasp-pages/directory-listing.html`	Go template used to render directory listings.

16.2.1 Directory Listing Template

When `enable_directory_listing = true` in `config/axonasp.toml`, the server renders a directory listing using the HTML template at `directory_listing_template`. The default template references `www/axonasp-pages/directory-listing.html`.

```
[server]
enable_directory_listing = true
directory_listing_template = "./www/axonasp-pages/directory-listing.html"
```

The template is a standard Go `html/template` file. It receives the following data context:

Variable	Type	Description
`.Title`	string	Page title, typically the directory path.
`.RequestPath`	string	The URL path of the requested directory.
`.ParentPath`	string	URL path to the parent directory. Empty for the web root.
`.LogoDataURI`	string	Base64-encoded data URI for the AxonASP logo image.
`.InlineCSS`	string	Full CSS content inlined into the `<style>` tag.
`.Entries`	slice	List of directory entries (files and subdirectories).

Each `.Entries` item exposes:

Field	Type	Description
`.Name`	string	File or directory name.
`.Href`	string	URL link for the entry.
`.TypeLabel`	string	`File` or `Directory`.
`.MimeType`	string	Detected MIME type for files; `-` for directories.
`.SizeDisplay`	string	Human-readable file size; `-` for directories.
`.Modified`	string	Last-modified timestamp as a formatted string.

Customization: You can replace the default template with your own by pointing `directory_listing_template` to a different file path. The template must use standard Go `html/template` syntax and the variable names listed above.

16.2.2 Blocking Direct Access

Both `www/axonasp-pages/` and `www/error-pages/` are listed in `blocked_dirs` in `config/axonasp.toml`:

```
blocked_dirs = [
  "./www/error-pages",
  "./www/axonasp-pages",
]
```

This setting prevents clients from accessing these directories directly via HTTP. Requests to paths under these directories return a 404 response.

16.3 error-pages/

Path: `www/error-pages/`

This directory stores custom HTTP error pages served by the AxonASP HTTP server. When a request results in an HTTP error code, the server checks this directory for a matching error page before using its built-in error response.

16.3.1 Naming Convention

Error pages are matched by filename:

Filename pattern	Description

Filename pattern	Description
`<status_code>.asp`	An ASP script executed dynamically for the given status code.
`<status_code>.html`	A static HTML file served for the given status code.

Examples: - `403.html` — Served on HTTP 403 Forbidden responses. - `404.html` — Served on HTTP 404 Not Found responses. - `500.html` — Served on HTTP 500 Internal Server Error responses. - `500.asp` — An ASP page executed and served on HTTP 500 responses.

When both an `.asp` and an `.html` file exist for the same status code, the `.asp` file takes precedence.

16.3.2 Default Error Pages

AxonASP ships with default static error pages for the most common status codes (403, 404, 500). You can replace these files with your own pages by editing the files in `www/error-pages/`.

16.3.3 Using web.config for Custom Error Mapping

You can also map status codes to custom pages using the `<httpErrors>` section in `web.config`. This approach allows you to use pages outside the `error-pages/` directory and to specify whether the page is executed dynamically or served as a static file:

```
<configuration>
  <system.webServer>
    <httpErrors>
      <error statusCode="404" path="/error-pages/404.html" responseMode="
File" />
      <error statusCode="500" path="/error-pages/500.asp" responseMode="E
xecuteURL" />
    </httpErrors>
  </system.webServer>
</configuration>
```

16.4 Remarks

- Direct HTTP access to both `www/axonasp-pages/` and `www/error-pages/` is blocked by default. Do not remove these directories from `blocked_dirs` in `axonasp.toml`.
- The directory-listing template is loaded from disk on each request when directory listing is triggered. CSS and logo assets are inlined into the HTTP response; they are not served as separate requests.

- Error pages placed in `www/error-pages/` have access to all standard ASP intrinsic objects (`Request`, `Response`, `Server`, `Session`, `Application`) when using `.asp` format.
- The `axonasp-pages/css/` stylesheet is also used by the built-in documentation manual. Modifications to the stylesheet affect both the manual appearance and any custom pages that reference it.

17 AxonASP Configuration Reference

17.1 Overview

The `axonasp.toml` file contains all configuration settings for the AxonASP server. This file is shared across all runtime modes: HTTP server, FastCGI server, CLI, and MCP server. All settings can be overridden using environment variables, making it easy to manage different configurations for development and production environments.

17.2 Configuration Location

The configuration file should be located at:

`./config/axonasp.toml`

Both the configuration folder and web root (`./www/`) must be in the same directory as the executable (or use absolute paths).

17.3 Environment Variable Override

All configuration values can be overridden via environment variables when `viper_automatic_env = true` in the configuration:

Format: `SECTION_SETTING` or `SECTION_SUBSETTING` (uppercase with underscores replacing dots)

Examples:

```
# Override global settings
$env:DEFAULT_CHARSET = "UTF-8"
$env:DEFAULT_SCRIPT_TIMEOUT = "120"

# Override server settings
$env:SERVER_PORT = "8802"
$env:WEB_ROOT = "C:\myapp\www"

# Override FastCGI settings
$env:FASTCGI_SERVER_PORT = "9001"

# Override database settings (G3DB)
$env:MYSQL_HOST = "db.example.com"
```

```
$env:MYSQL_USER = "appuser"
$env:MYSQL_PASS = "securepassword"
```

Using .env File:

Create a `.env` file in the same directory as the executable:

```
DEFAULT_CHARSET=UTF-8
SERVER_PORT=8802
MYSQL_HOST=db.example.com
MYSQL_USER=appuser
MYSQL_PASS=securepassword
```

17.4 Global Settings `[global]`

Global settings apply to all AxonASP runtime modes.

17.4.1 default_charset

Type: String
Default: `"UTF-8"`
Environment Variable: `DEFAULT_CHARSET`

Sets the character encoding used when serving content. Controls the `Content-Type: text/html; charset=` HTTP header.

Notes: - Only UTF-8 is supported internally; this setting is for compatibility with legacy ASP applications - HTTP server and FastCGI server include this charset in response headers - CLI always uses UTF-8 regardless of this setting

Example:

```
default_charset = "UTF-8"
```

17.4.2 default_mslcid

Type: Integer (LCID - Locale ID)
Default: `1033` (English - United States)
Environment Variable: `DEFAULT_MSLCID`

Sets the default locale for ASP applications. Affects date/time formatting, number formatting, and language-specific functions.

Common LCIDs: - `1033` - English (United States) - `2057` - English (United Kingdom) - `1046` - Portuguese (Brazil) - `1051` - German (Germany) - `1035` - Finnish - `1040` - Italian - `3082` - Spanish (Spain) - `1036` - French (France)

Example:

```
default_mslcid = 1046  # Portuguese (Brazil)
```

17.4.3 default_script_timeout

Type: Integer (seconds)
Default: `60`
Environment Variable: `DEFAULT_SCRIPT_TIMEOUT`

Maximum execution time before the server terminates an ASP script. Prevents runaway scripts from consuming resources indefinitely.

Guidelines: - **Development:** 120-300 seconds (allow for interactive debugging) - **Production:** 30-60 seconds (strict resource control) - **Long-Running Tasks:** 300+ seconds (but consider background jobs instead)

Example:

```
default_script_timeout = 60
```

17.4.4 response_buffer_limit_mb

Type: Integer (megabytes)
Default: 4
Environment Variable: `RESPONSE_BUFFER_LIMIT_MB`

Maximum size of buffered output when `Response.Buffer` is enabled before aborting with a runtime error. Protects against unbounded memory consumption.

Guidelines: - **Small Sites:** 2-4 MB (default) - **Medium Sites:** 8-16 MB - **Large Sites:** 32-64 MB (requires proportional memory allocation)

Example:

```
response_buffer_limit_mb = 4
```

17.4.5 default_timezone

Type: String (IANA timezone)
Default: `"UTC"`
Environment Variable: `DEFAULT_TIMEZONE`

Default timezone for the server. Affects date/time calculations and conversions in ASP applications.

Common Timezones: - `"UTC"` - `"America/New_York"` - `"America/Chicago"` - `"America/Los_Angeles"` - `"Europe/London"` - `"Europe/Paris"` - `"Asia/Tokyo"` - `"Asia/Shanghai"` - `"Australia/Sydney"` - `"America/Sao_Paulo"`

Example:

```
default_timezone = "America/New_York"
```

17.4.6 enable_asp_debugging

Type: Boolean
Default: `true`
Environment Variable: `ENABLE_ASP_DEBUGGING`

When enabled, the HTTP/FastCGI server provides detailed error messages, stack traces, and debugging information. Helpful during development but exposes sensitive information.

⚠️ **Security Warning:** Disable in production environments!

Example:

```
enable_asp_debugging = true  # Development only!
```

17.4.7 enable_error_log_file

Type: Boolean
Default: `true`
Environment Variable: `ENABLE_ERROR_LOG_FILE`

When enabled, creates an `error.log` file in `./temp/` directory containing error details and stack traces. Useful for troubleshooting production issues.

Example:

```
enable_error_log_file = true
```

17.4.8 dump_preprocessed_source

Type: Boolean
Default: `false`
Environment Variable: `DUMP_PREPROCESSED_SOURCE`

When enabled, saves the preprocessed ASP source code to `./temp/` before compilation. Useful for debugging parser issues but consumes disk space on high-traffic sites.

⚠️ **Performance Warning:** Disable in production! Can create thousands of temporary files.

Example:

```
dump_preprocessed_source = false
```

17.4.9 clean_sessions_on_startup

Type: Boolean
Default: `true`
Environment Variable: `CLEAN_SESSIONS_ON_STARTUP`

When enabled, clears all existing sessions when the server starts. Prevents stale session data but logs out active users.

Use Cases: - `true` - Development, testing, or security-critical restarts - `false` - Production with persistent user sessions

Example:

```
clean_sessions_on_startup = false  # Preserve sessions across restarts
```

17.4.10 bytecode_caching_enabled

Type: String (Enum)
Default: `"enabled"`
Environment Variable: `BYTECODE_CACHING_ENABLED`
Valid Values: `"enabled"`, `"memory-only"`, `"disk-only"`, `"disabled"`

Controls compiled bytecode caching strategy for ASP scripts: - `"enabled"` - Cache in both memory (tier 1) and disk (tier 2) for maximum performance - `"memory-only"` - Cache only in RAM (tier 1); faster with higher memory usage - `"disk-only"` - Cache only on disk (tier 2); slower but lower memory footprint - `"disabled"` - No caching; recompile every request (development/debugging only)

Performance Impact:

```
enabled > memory-only > disk-only > disabled
```

Example:

```
bytecode_caching_enabled = "enabled"
```

17.4.11 cache_max_size_mb

Type: Integer (megabytes)
Default: 128
Environment Variable: `CACHE_MAX_SIZE_MB`

Maximum size of the compiled bytecode cache in memory. When exceeded, least-recently-used scripts are evicted.

Guidelines: - **Small Sites:** 64-128 MB - **Medium Sites:** 128-256 MB - **Large Sites:** 256-512 MB (requires proportional system RAM)

Example:

```
cache_max_size_mb = 128
```

17.4.12 clean_cache_on_startup

Type: Boolean
Default: `true`
Environment Variable: `CLEAN_CACHE_ON_STARTUP`

When enabled, clears the compiled script cache on server startup. Ensures new builds are executed but increases startup time.

Guidelines: - `true` - Development (pick up recent changes) - `false` - Production (faster startup)

Example:

```
clean_cache_on_startup = true
```

17.4.13 vm_pool_size

Type: Integer (0 = unlimited)
Default: `100`
Environment Variable: `VM_POOL_SIZE`

Number of virtual machine instances in the execution pool. Each VM can execute one script simultaneously, so pool size determines concurrent execution capacity.

Guidelines: - **Small Sites:** 25-50 VMs - **Medium Sites:** 50-100 VMs - **High-Traffic Sites:** 100-200+ VMs (0 for unlimited)

⚠️ Setting too high causes memory exhaustion; too low causes request queueing.

Example:

```
vm_pool_size = 100
```

17.4.14 golang_memory_limit_mb

Type: Integer (0 = unlimited)
Default: `128`
Environment Variable: `GOLANG_MEMORY_LIMIT_MB`

Maximum memory the Go runtime can allocate (in megabytes). `0` means unlimited.

Guidelines: - **Development:** 128-256 MB - **Production:** 256-512 MB (or higher based on workload) - **Containerized:** Set based on container limits

⚠️ Note: Go runtime may not strictly enforce this limit; actual usage can vary.

Example:

```
golang_memory_limit_mb = 512
```

17.4.15session_flush_interval_seconds

Type: Integer (seconds)
Default: 15
Environment Variable: SESSION_FLUSH_INTERVAL_SECONDS

Interval for asynchronously flushing dirty in-memory sessions to disk. Higher values reduce disk I/O; lower values provide fresher data.

Guidelines: - 0 - Synchronous writes (safe but slower) - 5-10 - Frequent writes (good for critical data) - 15-30 - Balanced (default) - 60+ - Reduced I/O (requires crash-safe handling)

Example:

```
session_flush_interval_seconds = 15
```

17.4.16 adodb_platform_architecture

Type: String (Enum)
Default: "auto"
Environment Variable: ADODB_PLATFORM_ARCHITECTURE
Valid Values: "auto", "amd64", "386"

Specifies platform architecture for ADODB Access database driver (Windows only).

- "auto" - Automatically detect platform architecture
- "amd64" - 64-bit (modern systems)
- "386" - 32-bit (legacy systems)

Example:

```
adodb_platform_architecture = "auto"
```

17.4.17execute_as_asp

Type: Array of Strings
Default: [".asp"]
Environment Variable: EXECUTE_AS_ASP (comma-separated list)

File extensions to execute as ASP scripts. Other extensions are served as static files.

Examples:

```
# Default: only .asp files
execute_as_asp = [".asp"]
```

```
# Multiple formats
execute_as_asp = [".asp", ".aspx", ".cer", ".asa"]
```

17.4.18 viper_watch_config

Type: Boolean
Default: `false`
Environment Variable: `VIPER_WATCH_CONFIG`

When enabled, automatically reloads configuration file on changes without restarting the server.

⚠️ Note: Not fully implemented; manual restart currently required for most changes.

Example:

```
viper_watch_config = false
```

17.4.19 viper_automatic_env

Type: Boolean
Default: `true`
Environment Variable: `VIPER_AUTOMATIC_ENV`

When enabled, automatically reads configuration values from environment variables. Allows environment variables to override `axonasp.toml` settings.

Example:

```
viper_automatic_env = true
```

17.5 CLI Settings [cli]

Configuration for command-line interface (`axonasp-cli.exe`).

17.5.1 enable_cli

Type: Boolean
Default: `true`
Environment Variable: `ENABLE_CLI`

When enabled, allows the TUI (Text User Interface) for interactive ASP script testing and execution. Required for CLI to function.

Example:

```
enable_cli = true
```

17.5.2 enable_cli_run_from_command_line

Type: Boolean
Default: `true`
Environment Variable: `ENABLE_CLI_RUN_FROM_COMMAND_LINE`

When enabled, allows running ASP scripts directly from command line:

```
.\axonasp-cli.exe -r script.asp
```

⚠️ Security: Only enable if you trust the scripts being executed.

Example:

```
enable_cli_run_from_command_line = true
```

17.6 Web Server Settings `[server]`

Configuration for HTTP web server (`axonasp-http.exe`).

17.6.1 default_error_pages_directory

Type: String (path)
Default: `"./www/error-pages"`
Environment Variable: `DEFAULT_ERROR_PAGES_DIRECTORY`

Directory containing custom error page templates (e.g., `404.asp`, `500.asp`).

Files Searched: - `404.asp` / `404.html` - Not Found - `500.asp` / `500.html` - Internal Server Error - `403.asp` / `403.html` - Forbidden - `400.asp` / `400.html` - Bad Request

Example:

```
default_error_pages_directory = "./www/error-pages"
```

17.6.2 web_root

Type: String (path)
Default: `"./www/"`
Environment Variable: `WEB_ROOT`

Root directory for serving files. All requests are resolved relative to this path.

Security: This setting cannot be overridden by `web.config` files.

Example:

```
web_root = "./www/"
```

17.6.3 default_pages

Type: Array of Strings
Default: `["index.asp", "default.asp", "index.html", ...]`
Environment Variable: `DEFAULT_PAGES` (comma-separated)

Default pages to serve when a directory is requested. Server tries each in order; serves the first found.

Example:

```
default_pages = [
  "index.asp",
  "default.asp",
  "index.html",
  "default.html",
]
```

17.6.4 server_port

Type: Integer or String
Default: `8801`
Environment Variable: `SERVER_PORT`

Port on which the HTTP server listens. Point your reverse proxy to this port.

⚠️ Recommended: Expose only through reverse proxy (nginx, Apache, IIS, Caddy)

Example:

```
server_port = 8801
```

17.6.5 blocked_extensions

Type: Array of Strings
Default: `[".asax", ".ascx", ".config", ".exe", ".dll", ...]`
Environment Variable: `BLOCKED_EXTENSIONS` (comma-separated)

File extensions that cannot be served by the web server (returns 404).

Security Best Practices: - Block source code: `.cs`, `.vb`, `.asp`, `.asa` - Block configuration: `.config`, `.toml`, `.env` - Block binaries: `.exe`, `.dll`, `.so` - Block database: `.mdb`, `.db`, `.sqlite`

Example:

```
blocked_extensions = [
  ".asa",
  ".cer",
  ".config",
```

```
  ".cs",
  ".exe",
  ".dll",
  ".env",
  ".mdb",
  ".toml",
  ".vb",
]
```

17.6.6 blocked_files

Type: Array of Strings
Default: `["MyInfo.xml"]`
Environment Variable: `BLOCKED_FILES` (comma-separated)

Specific files that cannot be served (returns 404).

Example:

```
blocked_files = [
  "MyInfo.xml",
  "web.config",
  "global.asa",
]
```

17.6.7 blocked_dirs

Type: Array of Strings
Default: `["./www/error-pages", "./www/axonasp-pages"]`
Environment Variable: `BLOCKED_DIRS` (comma-separated)

Directories that cannot be accessed directly (returns 404).

Example:

```
blocked_dirs = [
  "./www/error-pages",
  "./www/axonasp-pages",
  "./www/config",
]
```

17.6.8 enable_webconfig

Type: Boolean
Default: `true`
Environment Variable: `ENABLE_WEBCONFIG`

When enabled, reads `web.config` files from directories and applies settings. Allows per-application configuration (custom error pages, redirects, etc.).

Example:

```
enable_webconfig = true
```

17.6.9 enable_directory_listing

Type: Boolean
Default: `true`
Environment Variable: `ENABLE_DIRECTORY_LISTING`

When enabled, displays directory contents if no default page is found (security risk in production).

⚠️ Disable in production!

Example:

```
enable_directory_listing = false  # Production setting
```

17.6.10 directory_listing_template

Type: String (path)
Default: `"./www/axonasp-pages/directory-listing.html"`
Environment Variable: `DIRECTORY_LISTING_TEMPLATE`

HTML template used for directory listing UI. Can be customized to match site design.

Example:

```
directory_listing_template = "./www/axonasp-pages/directory-listing.html"
```

17.7 FastCGI Server Settings `[fastcgi]`

Configuration for FastCGI application server (`axonasp-fastcgi.exe`).

17.7.1 default_pages

Type: Array of Strings
Environment Variable: `FASTCGI_DEFAULT_PAGES` (comma-separated)

Default pages for FastCGI mode (same purpose as HTTP server).

Example:

```
[fastcgi]
default_pages = [
  "index.asp",
  "default.asp",
```

```
  "index.html",
]
```

17.7.2 server_port

Type: Integer or String
Default: `9000`
Environment Variable: `FASTCGI_SERVER_PORT`

Port or socket for FastCGI listener. Can be TCP port or Unix socket path.

TCP Examples:

```
server_port = 9000
```

Unix Socket Examples (Linux/macOS):

```
server_port = "unix:/tmp/axonasp.sock"
```

17.8 Database Configuration `[g3db]`

Configuration for G3DB library (multi-database support).

17.8.1 MySQL Settings

```
[g3db]
mysql_host = "localhost"
mysql_port = 3306
mysql_user = "root"
mysql_pass = "password"
mysql_database = "test"
```

Environment Variables: - `MYSQL_HOST` - `MYSQL_PORT` - `MYSQL_USER` - `MYSQL_PASS` - `MYSQL_DATABASE`

17.8.2 PostgreSQL Settings

```
[g3db]
postgres_host = "localhost"
postgres_port = 5432
postgres_user = "postgres"
postgres_pass = "password"
postgress_database = "test"
postgress_ssl_mode = "disable"
```

Environment Variables: - `POSTGRES_HOST` - `POSTGRES_PORT` - `POSTGRES_USER` - `POSTGRES_PASS` - `POSTGRESS_DATABASE` - `POSTGRESS_SSL_MODE`

17.8.3 MS SQL Server Settings

```
[g3db]
mssql_host = "localhost"
mssql_port = 1433
mssql_user = "sa"
mssql_pass = "password"
mssql_database = "test"
```

Environment Variables: - MSSQL_HOST - MSSQL_PORT - MSSQL_USER - MSSQL_PASS - MSSQL_DATABASE

17.8.4 SQLite Settings

```
[g3db]
sqlite_path = "./database.db"
sqlite_busy_timeout = 5000
```

Environment Variables: - SQLITE_PATH - SQLITE_BUSY_TIMEOUT (milliseconds)

17.8.5 Oracle Settings

```
[g3db]
# Option 1: Full DSN (takes precedence)
oracle_dsn = "oracle://user:password@localhost:1521/ORCLCDB"

# Option 2: Individual settings
oracle_host = "localhost"
oracle_port = 1521
oracle_user = "system"
oracle_pass = "password"
oracle_service = "ORCLCDB"
```

Environment Variables: - ORACLE_DSN - ORACLE_HOST - ORACLE_PORT - ORACLE_USER - ORACLE_PASS - ORACLE_SERVICE

17.9 Mail Configuration [g3mail]

Configuration for G3Mail library (email sending).

17.9.1 g3mail Settings

```
[g3mail]
smtp_host = "smtp.gmail.com"
smtp_port = 587
smtp_user = "your_email@gmail.com"
smtp_pass = "your_app_password"
smtp_from = "sender@example.com"
```

Environment Variables: - SMTP_HOST - SMTP_PORT - SMTP_USER - SMTP_PASS - SMTP_FROM

Gmail Configuration Example:

```
[g3mail]
smtp_host = "smtp.gmail.com"
smtp_port = 587  # Or 465 for TLS
smtp_user = "your-email@gmail.com"
smtp_pass = "your-app-specific-password"  # Not your account password!
smtp_from = "noreply@example.com"
```

⚠️ Security: Use environment variables or `.env` file for credentials, not hardcoded values.

17.10 AxonASP Functions Settings `[axfunctions]`

Configuration for built-in Ax functions.

17.10.1enable_global_ax

Type: Boolean
Default: `true`
Environment Variable: `ENABLE_GLOBAL_AX`

Makes Ax functions available globally without `Server.CreateObject()`.

Default Method (with setting disabled):

```
Set ax = Server.CreateObject("G3AXON.Functions")
result = ax.SomeFuction()
```

When enabled:

```
result = AxSomeFunction()  ' Direct call
```

⚠️ Breaks Classic ASP compatibility; enable only if needed.

Example:

```
enable_global_ax = true
```

17.10.2 enable_axservershutdown_function

Type: Boolean
Default: `false`
Environment Variable: `ENABLE_AXSERVERSHUTDOWN_FUNCTION`

When enabled, allows ASP scripts to call `AxShutdownAxonASPServer()` to shut down the server programmatically.

⚠️ **Security Warning:** Disable in production! Prevents unauthorized server shutdowns.

Example:

```
enable_axservershutdown_function = false
```

17.10.3 ax_default_css_path

Type: String (path)
Default: `"./www/axonasp-pages/css/axonasp.css"`
Environment Variable: `AX_DEFAULT_CSS_PATH`

CSS file used for built-in AxonASP pages (error pages, etc.).

Example:

```
ax_default_css_path = "./www/axonasp-pages/css/axonasp.css"
```

17.10.4 ax_default_logo_path

Type: String (path)
Default: `"./www/axonasp-pages/images/logo.png"`
Environment Variable: `AX_DEFAULT_LOGO_PATH`

Logo file for built-in AxonASP pages (served as inline base64 image).

Example:

```
ax_default_logo_path = "./www/axonasp-pages/images/logo.png"
```

17.11 MCP Server Settings `[mcp]`

Configuration for Model Context Protocol server (`axonasp-mcp.exe`).

17.11.1 mcp_mode

Type: String (Enum)
Default: `"stdio"`
Environment Variable: `MCP_MODE`
Valid Values: `"stdio"`, `"sse"`

Communication mode for MCP: - `"stdio"` - Standard input/output (local/CLI usage) - `"sse"` - Server-Sent Events (remote web access)

Example:

```
mcp_mode = "stdio"
```

17.11.2 mcp_sse_port

Type: Integer
Default: `8000`
Environment Variable: `MCP_SSE_PORT`

Port for SSE server when `mcp_mode` is set to `"sse"`. Ignored in `"stdio"` mode.

Access Points: - SSE stream: `http://localhost:8000/sse` - Command API: `http://localhost:8000/command`

Example:

```
mcp_sse_port = 8000
```

17.11.3 mcp_docs

Type: String (path)
Default: `"mcp/docs.md"`
Environment Variable: `MCP_DOCS`

Path to markdown documentation file used by MCP for query responses and assistance.

Example:

```
mcp_docs = "mcp/docs.md"
```

17.12 MSWC Component Settings `[mswc]`

Configuration for Microsoft Web Components (MSWC) like PageCounter.

17.12.1 pagecounter_enabled

Type: Boolean
Default: `false`
Environment Variable: `PAGECOUNTER_ENABLED`

When enabled, provides MSWC.PageCounter component for tracking page hit counts.

Example:

```
pagecounter_enabled = false
```

17.12.2 pagecounter_file

Type: String (path)
Default: `"./temp/hitcnt.gob"`
Environment Variable: `PAGECOUNTER_FILE`

Binary file (Go gob format) storing page hit counts. Created automatically if missing.

Example:

```
pagecounter_file = "./temp/hitcnt.gob"
```

17.12.3 pagecounter_save_interval_seconds

Type: Integer (seconds)
Default: `120`
Environment Variable: `PAGECOUNTER_SAVE_INTERVAL_SECONDS`

Interval for flushing hit counts to disk. Balances performance vs. data freshness.

Guidelines: - Lower values ⇢ More frequent writes ⇢ Fresher data but higher I/O - Higher values ⇢ Less frequent writes ⇢ Better performance but data loss on crash

Example:

```
pagecounter_save_interval_seconds = 120
```

17.13 Service Wrapper Settings [service]

Configuration for the service wrapper binary (axonasp-service on Unix and axonasp-service.exe on Windows).

This mode is designed for easier setup, mainly on Windows and small installations. For advanced Unix production deployments, prefer native service manager configuration from the runtime service pages.

17.13.1 service_name

Type: String
Default: “AxonASPServer”
Environment Variable: SERVICE_SERVICE_NAME

Internal service identifier used by the operating system service manager.

Example:

```
[service]
service_name = "AxonASPServer"
```

17.13.2 service_display_name

Type: String
Default: “G3pix AxonASP Server”
Environment Variable: SERVICE_SERVICE_DISPLAY_NAME

Human-readable service name shown in service management tools.

Example:

```
[service]
service_display_name = "G3pix AxonASP Server"
```

17.13.3service_description

Type: String
Default: “AxonASP Service running AxonASP Server. This is a wrapper used by axonasp-http or axonasp-fastcgi.”
Environment Variable: SERVICE_SERVICE_DESCRIPTION

Service description shown by the service manager.

Example:

```
[service]
service_description = "AxonASP Service running AxonASP Server. This is a
wrapper used by axonasp-http or axonasp-fastcgi."
```

17.13.4 service_executable_path

Type: String (path)
Default: “./axonasp-http”
Environment Variable: SERVICE_SERVICE_EXECUTABLE_PATH

Executable path started by the wrapper process.

Behavior:

- Relative paths are resolved from the wrapper executable directory.
- Windows automatically appends .exe when no file extension is provided.
- Can target axonasp-http or axonasp-fastcgi.

Example:

```
[service]
service_executable_path = "./axonasp-fastcgi"
```

17.13.5 service_environment_variables

Type: Array of Strings (KEY=VALUE format)
Default: []
Environment Variable: SERVICE_SERVICE_ENVIRONMENT_VARIABLES

Environment variable entries applied to the child runtime process.

Example:

```
[service]
service_environment_variables = ["SERVER_SERVER_PORT=9901", "GLOBAL_DEFAU
LT_TIMEZONE=UTC"]
```

17.14 Configuration Examples

17.14.1 Development Setup

```
[global]
default_charset = "UTF-8"
default_mslcid = 1033
default_script_timeout = 300
enable_asp_debugging = true
enable_error_log_file = true
dump_preprocessed_source = false
clean_sessions_on_startup = false
bytecode_caching_enabled = "enabled"
cache_max_size_mb = 256
clean_cache_on_startup = true
vm_pool_size = 50
golang_memory_limit_mb = 256

[server]
server_port = 8801
enable_directory_listing = true
```

17.14.2 Production Setup

```
[global]
default_charset = "UTF-8"
default_mslcid = 1033
default_script_timeout = 60
enable_asp_debugging = false
enable_error_log_file = true
dump_preprocessed_source = false
clean_sessions_on_startup = false
bytecode_caching_enabled = "enabled"
cache_max_size_mb = 512
clean_cache_on_startup = false
vm_pool_size = 200
```

```
golang_memory_limit_mb = 512

[server]
server_port = 8801
enable_directory_listing = false

[cli]
enable_cli = false
enable_cli_run_from_command_line = false
```

17.14.3 Docker/Container Setup

Use environment variables instead of modifying the configuration file:

```yaml
# docker-compose.yml example
services:
  axonasp:
    image: axonasp:latest
    environment:
      DEFAULT_SCRIPT_TIMEOUT: "120"
      SERVER_PORT: "8801"
      VM_POOL_SIZE: "100"
      GOLANG_MEMORY_LIMIT_MB: "256"
      MYSQL_HOST: "mysql"
      MYSQL_USER: "appuser"
      MYSQL_PASS: "securepassword"
    ports:
      - "8801:8801"
```

17.15 Best Practices

1. **Development vs. Production:** Use different configurations or environment variables per environment
2. **Security:** Never hardcode credentials; use environment variables or `.env` files
3. **Performance:** Enable bytecode caching and set appropriate pool sizes
4. **Debugging:** Enable debugging/logging only during development
5. **Backups:** Keep backups of your configuration file before making changes
6. **Monitoring:** Monitor cache hit rates, error logs, and VM pool usage
7. **Scaling:** For high traffic, use reverse proxy with multiple AxonASP instances

For more information, see Project Structure and deployment examples.

- viper_watch_config enables live reload hooks in host runtimes that support it.
- Keep production values strict for blocking rules, memory limits, and directory listing policy.

17.16 Code Example

```
[global]
default_script_timeout = 90
response_buffer_limit_mb = 8
bytecode_caching_enabled = "memory-only"
cache_max_size_mb = 256
vm_pool_size = 150
golang_memory_limit_mb = 512
enable_asp_debugging = false
enable_error_log_file = true

enable_webconfig = true
```

18 The #include Directive

18.1 Overview

The **#include** directive is a server-side include (SSI) mechanism used in G3Pix AxonASP to insert the content of one file into another before the script is executed by the Virtual Machine. This directive is essential for code reuse, allowing developers to maintain shared components such as headers, footers, database connection strings, and library functions in a single location.

18.2 Syntax

The directive must be placed inside HTML comments and uses either the **file** or **virtual** attribute:

```
<!--#include file="filename"-->
<!--#include virtual="/path/filename"-->
```

18.3 Parameters and Arguments

- **file** (String): Specifies a path relative to the directory of the current file. You cannot use absolute paths or "../" to move up in the directory structure with this attribute.
- **virtual** (String): Specifies a path relative to the web root of the AxonASP application. This is the preferred method for including files located in global or shared directories.

18.4 How it Works

The AxonASP Lexer identifies the **#include** directive during the initial parsing phase. The engine then retrieves the content of the specified file and physically injects it into the source stream, replacing the directive tag.

- **Preprocessing**: Includes are processed *before* any VBScript code is executed. You cannot use VBScript variables to dynamically set the path of an include directive.
- **Scope**: Variables, procedures, and classes defined in an included file are available to the parent script as if they were written directly within it.
- **Nesting**: Included files can themselves contain **#include** directives, allowing for deeply nested component architectures.
- **Circular References**: AxonASP prevents infinite recursion by limiting include depth and detecting circular references.

18.5 Remarks

- **File Extensions**: While `.inc` and `.asp` are the most common extensions for included files, AxonASP processes any file type specified in the directive as script content.
- **Performance**: Excessive use of deeply nested includes can impact initial compilation time, though AxonASP's high-performance script cache mitigates this for subsequent requests.
- **Alternative**: For dynamic file inclusion during runtime, use `Server.Execute` or `Server.Transfer`.

18.6 Code Example

The following example demonstrates a standard page structure using include directives for a consistent layout.

```
<!--#include virtual="/includes/config.inc"-->
<!--#include file="header.inc"-->

<%
' Main page logic
Response.Write "<h1>Welcome to the Dashboard</h1>"
Response.Write "<p>Connected to: " & DB_NAME & "</p>"
%>

<!--#include file="footer.inc"-->
```

19 ASP Variables

19.1 Overview

Variables in G3Pix AxonASP are used to store and manipulate data during script execution. In the AxonASP Virtual Machine, every variable is a **Variant**, a special data type that can hold different kinds of information, including strings, integers, floating-point numbers, booleans, dates, and object references.

19.2 Syntax

Variables are declared using the **Dim** statement and assigned values using the equals operator (=). Object references require the **Set** keyword.

```
Dim variableName
variableName = value

Set objectVariable = Server.CreateObject("ProgID")
```

19.3 How it Works

The AxonASP VM manages variables using an efficient, stack-based architecture. - **Variant Type**: The engine automatically handles type conversion (coercion) based on the operation performed. For example, adding a string that looks like a number to an integer will result in a numeric addition. - **Option Explicit**: This directive forces the explicit declaration of all variables. Using it is a foundational mandate in AxonASP to prevent bugs caused by mistyped variable names. - **Scope**: - **Local**: Declared within a procedure (Sub or Function); available only within that procedure. - **Global**: Declared outside procedures; available to all scripts in the same page and its includes. - **Object Lifecycle**: Variables holding object references must be explicitly cleared by setting them to **Nothing** to ensure immediate resource cleanup.

19.4 API Reference

19.4.1 Declarations

- **Dim**: The standard way to declare one or more variables.
- **Public**: Declares variables with global scope (used in classes or global modules).
- **Private**: Declares variables accessible only within the script or class where they are defined.
- **Const**: Declares a constant value that cannot be changed during execution.

19.4.2 Assignment

- **=**: Assigns a scalar value (String, Number, Boolean) to a variable.
- **Set**: Assigns an object reference to a variable.

19.5 Remarks

- **Naming Conventions**: Variable names must begin with an alphabetic character and cannot contain periods or spaces.
- **Initialization**: Variables are initialized to **Empty** by default. Use `IsEmpty()` to check this state.
- **Case Insensitivity**: Variable names in AxonASP are case-insensitive (`myVar` is the same as `MYVAR`).

- **No Inline Initialization**: You cannot declare and assign a variable on the same line (e.g., `Dim x = 10` is invalid).

19.6 Code Example

The following example demonstrates variable declaration, assignment of different types, and proper object handling.

```
<%
Option Explicit

' Scalar variables
Dim userName, userAge, isActive
userName = "Lucas"
userAge = 30
isActive = True

' Constant declaration
Const MAX_ATTEMPTS = 5

' Object variable
Dim dict
Set dict = Server.CreateObject("Scripting.Dictionary")
dict.Add "ID", 101
dict.Add "Status", "Active"

' Output values
Response.Write "User: " & userName & " (Age: " & userAge & ")<br>"
Response.Write "Dictionary Count: " & dict.Count

' Cleanup
Set dict = Nothing
%>
```

20 ASP Procedures

20.1 Overview

Procedures in G3Pix AxonASP are discrete blocks of code that perform specific tasks. They allow for code modularity, better organization, and the implementation of reusable logic. AxonASP supports two types of procedures: **Sub** (Subroutines), which perform actions without returning a value, and **Function**, which perform calculations and return a result to the caller.

20.2 Syntax

Procedures are defined using the **Sub** or **Function** keywords and must be explicitly closed with **End Sub** or **End Function**.

20.2.1 Subroutine

```
Sub SubName(arg1, arg2)
    ' logic here
End Sub
```

20.2.2 Function

```
Function FuncName(arg1, arg2)
    ' logic here
    FuncName = result ' Set return value
End Function
```

20.3 How it Works

The AxonASP Virtual Machine executes procedures using a localized stack frame. - **Parameters**: Arguments are passed by reference (**ByRef**) by default, meaning changes to the parameter within the procedure affect the original variable. Use the **ByVal** keyword to pass a copy of the value instead. - **Return Values**: To return a value from a function, assign the result to the function's name within its body. - **Call Syntax**: - **Subs**: Called without parentheses, or by using the **Call** keyword with parentheses. - **Functions**: Parentheses are required when the return value is being assigned to a variable. - **Recursion**: AxonASP supports recursive procedure calls with a safe stack limit.

20.4 API Reference

20.4.1 Procedure Types

- **Sub**: A block of code that performs an action. It cannot be used in expressions.
- **Function**: A block of code that returns a value. It can be used in expressions and assignments.

20.4.2 Keywords

- **ByRef**: (Default) Passes the memory address of the argument.
- **ByVal**: Passes a copy of the argument's value.
- **Exit Sub / Exit Function**: Immediately terminates execution of the procedure and returns control to the caller.

20.5 Remarks

- **Scope**: Procedures defined in the main page or its includes are globally accessible throughout that page's execution.
- **Parentheses Trap**: A common error is using parentheses when calling a Sub without the `Call` keyword (e.g., `MySub(x, y)`). This can cause VBScript to treat the arguments as an expression, leading to unexpected behavior or syntax errors.
- **Case Insensitivity**: Procedure names are case-insensitive.

20.6 Code Example

The following example demonstrates the definition and invocation of both a Sub and a Function.

```
<%
Option Explicit

' Define a Function to calculate a total with tax
Function GetTotal(price, taxRate)
    GetTotal = price * (1 + taxRate)
End Function

' Define a Sub to log an operation
Sub LogOperation(msg)
    Response.Write "[LOG]: " & msg & "<br>"
End Sub

' --- Main Execution ---
Dim netPrice, total
netPrice = 100.00

' Function call (parentheses required for assignment)
total = GetTotal(netPrice, 0.15)

' Sub calls (different valid syntaxes)
LogOperation "Calculation complete."
Call LogOperation("Final total: " & total)

Response.Write "Amount Due: " & FormatCurrency(total)
%>
```

21 ASP Conditionals

21.1 Overview

Conditionals in G3Pix AxonASP allow your script to make decisions and execute different blocks of code based on specific criteria. The AxonASP Virtual Machine supports traditional boolean logic and comparison operations, providing the primary flow control mechanisms for dynamic application logic.

21.2 Syntax

AxonASP supports two main conditional structures: **If...Then...Else** and **Select Case**.

21.2.1 If Statement

```
If condition Then
    ' Code for True
```

```
ElseIf otherCondition Then
    ' Code for other True
Else
    ' Default code
End If
```

21.2.2 Select Case Statement

```
Select Case variable
    Case value1
        ' Code for value1
    Case value2
        ' Code for value2
    Case Else
        ' Default code
End Select
```

21.3 How it Works

Conditionals evaluate expressions into boolean values (**True** or **False**). - **Comparison Operators**: Includes `=`, `<>`, `<`, `>`, `<=`, and `>=`. - **Logical Operators**: Includes `And`, `Or`, `Not`, and `Xor`. - **Short-Circuit Evaluation**: **WARNING**: VBScript does *not* support short-circuit evaluation. In an expression like `If A And B Then`, both `A` and `B` are always evaluated. If `B` relies on `A` being true (e.g., checking an object property after checking if it exists), you must use nested **If** statements to avoid runtime errors. - **Select Case Semantics**: The `Select Case` statement performs a series of equality comparisons. Unlike some other languages, once a matching case is executed, the entire block terminates; there is no "fall-through" behavior.

21.4 API Reference

21.4.1 Keywords

- **If...Then**: Starts a conditional block.
- **ElseIf**: Provides an alternative condition if previous ones are false.
- **Else**: Provides a catch-all block if no conditions are met.
- **End If**: Required to close a block **If** statement.
- **Select Case**: Starts a multi-way branch block.
- **Case**: Defines a value to match against the expression.
- **Case Else**: Optional catch-all for the **Select Case** block.

21.5 Remarks

- **Mandatory Block Syntax**: In AxonASP, you should always use the block `If...Then...End If` syntax. Single-line `If` statements are discouraged as they are prone to parsing errors in complex scripts.
- **Coercion**: Non-boolean values are coerced during evaluation. For example, the integer `0` is treated as **False**, and non-zero integers are treated as **True**.

- **Case Insensitivity**: String comparisons using = are case-insensitive by default in AxonASP.

21.6 Code Example

The following example demonstrates defensive nesting to handle the lack of short-circuiting, and a `Select Case` for status handling.

```
<%
Option Explicit

Dim user, status
Set user = Nothing ' Simulate uninitialized object
status = "PENDING"

' Defensive nesting (VBScript evaluates both sides of 'And')
If Not user Is Nothing Then
    If user.IsActive Then
        Response.Write "User is active.<br>"
    End If
Else
    Response.Write "No user object found.<br>"
End If

' Status handling with Select Case
Select Case status
    Case "ACTIVE"
        Response.Write "Status: Green"
    Case "PENDING", "WAITING"
        Response.Write "Status: Amber"
    Case Else
        Response.Write "Status: Red"
End Select
%>
```

22 ASP Looping

22.1 Overview

Looping structures in G3Pix AxonASP allow you to execute a block of code multiple times. This is essential for iterating over arrays, processing records from a database, or performing repetitive calculations. AxonASP provides several types of loops to handle different scenarios, including fixed-count iterations and conditional repetitions.

22.2 Syntax

AxonASP supports **For...Next**, **For Each...Next**, **Do...Loop**, and **While...Wend** structures.

22.2.1 For Loop

```
For i = start To end [Step increment]
    ' code
Next
```

22.2.2 For Each Loop

```
For Each item In collection
    ' code
Next
```

22.2.3 Do Loop

```
Do While condition
    ' code
Loop

Do
    ' code
Loop Until condition
```

22.3 How it Works

The AxonASP Virtual Machine executes loop bodies repeatedly until a termination condition is met. - **For...Next**: Best for a known number of iterations. The `Step` keyword allows for increments other than 1. - **For Each...Next**: Specifically designed for iterating over collections (like the **ASP Dictionary** or **G3JSON** objects) and arrays. - **Do...Loop**: Highly flexible. Can check the condition at the beginning (`Do While`) or at the end (`Loop Until`), ensuring the code runs at least once in the latter case. - **Infinite Loops**: AxonASP includes safety mechanisms to terminate scripts that exceed the configured execution timeout, preventing infinite loops from hanging the server.

22.4 API Reference

22.4.1 Keywords

- **For** / **To** / **Next**: Defines a counter-based loop.
- **Each** / **In**: Defines a collection-based loop.
- **Do** / **While** / **Until** / **Loop**: Defines a condition-based loop.
- **While** / **Wend**: A legacy conditional loop (equivalent to `Do While...Loop`).
- **Exit For** / **Exit Do**: Immediately terminates the loop and moves execution to the statement following the loop closure.

22.5 Remarks

- **Loop Closures**: In AxonASP, `For` loops must end with `Next`, `Do` loops must end with `Loop`, and `While` loops must end with `Wend`.
- **Collection Performance**: `For Each` is generally faster and more readable when working with objects like the **Scripting.Dictionary**.

- **Counter Variable**: In `Next [variable]`, the variable name is optional but can improve readability in nested loops.

22.6 Code Example

The following example iterates over an array using a `For` loop and a dictionary using a `For Each` loop.

```
<%
Option Explicit

' Iterating over an Array
Dim colors, i
colors = Array("Red", "Green", "Blue")

Response.Write "<b>Colors:</b><br>"
For i = 0 To UBound(colors)
    Response.Write i & ": " & colors(i) & "<br>"
Next

' Iterating over a Dictionary
Dim dict, key
Set dict = Server.CreateObject("Scripting.Dictionary")
dict.Add "App", "AxonASP"
dict.Add "Engine", "Go"

Response.Write "<br><b>System Info:</b><br>"
For Each key In dict.Keys
    Response.Write key & " = " & dict(key) & "<br>"
Next

' Using Do Until
Dim count
count = 1
Do Until count > 3
    Response.Write "Loop count: " & count & "<br>"
    count = count + 1
Loop

Set dict = Nothing
%>
```

23 ASP Intrinsic Objects - Quick Reference

23.1 Overview

Classic ASP provides six built-in objects that are always available inside any ASP script without any `Server.CreateObject` call. These objects form the foundation of

every ASP application â€” they give your scripts access to the incoming HTTP request, the outgoing response, user sessions, application-wide state, server utilities, and error information.

AxonASP implements all six objects with full compatibility with the original IIS behavior.

23.2 At a Glance

Object	Purpose
Request	Reads data sent by the browser (query string, form fields, cookies, server variables)
Response	Writes output to the browser and controls the HTTP response (headers, cookies, redirects)
Session	Stores data for a single user across multiple page requests
Application	Stores data shared across all users and all requests for the application lifetime
Server	Utility methods â€” URL/HTML encoding, path translation, object creation, error info
ASPError	Provides structured error information when a script error occurs

---

23.3 Request

Represents data sent to the server by the client.

Key collections:

Collection	What it Contains
`Request.QueryString("key")`	URL query string parameters (`?name=value`)
`Request.Form("key")`	HTML form fields submitted via POST
`Request.Cookies("name")`	Client cookies
`Request.ServerVariables("var")`	HTTP headers and server environment info
`Request.ClientCertificate("key")`	Client certificate fields (HTTPS)

Collection	What it Contains
`Request("key")`	Searches all collections in order

Key properties: `Request.TotalBytes`
Key method: `Request.BinaryRead(n)`

23.4 Response

Controls what is sent back to the browser.

Key methods:

Method	Purpose
`Response.Write(text)`	Output text to browser
`Response.Redirect(url)`	302 redirect and stop
`Response.End()`	Immediately stop script execution
`Response.Flush()`	Send buffered content immediately
`Response.Clear()`	Clear the buffer
`Response.AddHeader(name, value)`	Add HTTP header
`Response.AppendToLog(message)`	Write to the server log
`Response.BinaryWrite(data)`	Send raw byte array

Key properties: `Response.Buffer`, `Response.ContentType`, `Response.Status`, `Response.Charset`, `Response.CacheControl`, `Response.Expires`, `Response.ExpiresAbsolute`, `Response.Cookies`, `Response.IsClientConnected`

23.5 Session

Stores user-specific data between requests using a cookie-based session ID.

Usage:

```
Session("UserName") = "Alice"
Dim name : name = Session("UserName")
Session.Abandon
```

Key properties: `Session.SessionID`, `Session.Timeout`, `Session.LCID`, `Session.CodePage`
Key methods: `Session.Abandon()`
Key collections: `Session.Contents`, `Session.StaticObjects`

23.6 Application

Application-level data store shared among all users. Must use `Lock/Unlock` to prevent race conditions.

Usage:

```
Application.Lock
Application("HitCount") = Application("HitCount") + 1
Application.Unlock
```

Key methods: `Application.Lock()`, `Application.Unlock()`
Key collections: `Application.Contents`, `Application.StaticObjects`

23.7 Server

Provides utility methods used throughout scripts.

Key methods:

Method	Purpose
`Server.CreateObject(progID)`	Instantiate a COM/native object
`Server.MapPath(path)`	Translate virtual path to physical path
`Server.HTMLEncode(str)`	Escape HTML special characters
`Server.URLEncode(str)`	Encode string for use in a URL query
`Server.URLPathEncode(str)`	Encode URL path segments
`Server.GetLastError()`	Return the current ASPError object
`Server.Execute(path)`	Execute another ASP file inline
`Server.Transfer(path)`	Transfer execution to another page

Key property: `Server.ScriptTimeout`

23.8 ASPError

Returned by `Server.GetLastError()` after an error occurs.

Key properties: `ASPCode`, `Number`, `Source`, `Description`, `File`, `Line`, `Column`, `Category`

23.9 Detailed Pages

- Request Object
- Response Object
- Session Object
- Application Object
- Server Object
- ASPError Object

24 ASP Quick Reference

24.1 Intrinsic Objects

- Request: read client input data (query, form, cookies, server variables)
- Response: write output, headers, cookies, redirects
- Server: utility methods and object creation
- Session: per-user state across requests
- Application: global state shared by all requests
- ASPError: structured error information

24.2 Runtime Pattern

1. Read inputs from Request
2. Apply validation and business logic
3. Use libraries through Server.CreateObject
4. Write output through Response
5. Persist state only when needed (Session/Application)

24.3 Minimal ASP Page

```
<%
Option Explicit
Dim name
name = Request.QueryString("name")
If Len(Trim(name)) = 0 Then
    name = "Guest"
End If
Response.ContentType = "text/html"
Response.Write "<h1>Hello, " & Server.HTMLEncode(name) & "</h1>"
%>
```

25 Write ASP Script Syntax

25.1 Overview

Classic ASP pages in G3Pix AxonASP combine HTML markup with server-side VBScript. Code inside <% and %> runs on the server before the response is sent to the client.

Use this page as a baseline syntax reference for script blocks, output, directives, comments, and control flow.

25.2 Prerequisites

- Save the page with `.asp` extension.
- Keep directives at the top of the file.
- Use VBScript syntax consistently in server blocks.

25.3 Core Delimiters

25.3.1 Execute server-side code

```
<%
Dim message
message = "Hello from server code"
Response.Write message
%>
```

25.3.2 Output an expression directly

```
<p>Current date: <%= Date() %></p>
```

25.3.3 Return value

- `<% ... %>` executes statements and returns no direct inline value.
- `<%= expression %>` evaluates `expression` and writes its String representation to the response output stream.

25.4 Write Output with Response.Write

25.4.1 Syntax

```
Response.Write expression
```

25.4.2 Return value

Returns no value. The method appends the evaluated text to the HTTP response body.

25.4.3 Example

```
<%
Response.Write "<h1>Dashboard</h1>"
Response.Write "<p>Welcome, " & Server.HTMLEncode("alice") & "</p>"
%>
```

25.5 Use Page Directives

Place directives at the top of the file.

```
<%@Language="VBSCRIPT" CodePage="65001"%>
<%
Response.CodePage = 65001
Response.Charset = "utf-8"
%>
```

25.5.1 Return value

Directives do not return values. They configure how AxonASP parses and executes the page.

25.6 Add Comments

```
<%
' Single-line VBScript comment
Response.Write "Comments are ignored by the runtime"
%>
```

25.6.1 Return value

Comments do not return values and do not produce output.

25.7 Basic Control Flow

25.7.1 If…Then…Else

```
<%
Dim isAdmin
isAdmin = True

If isAdmin Then
    Response.Write "<p>Admin panel enabled.</p>"
Else
    Response.Write "<p>Standard user mode.</p>"
End If
%>
```

25.7.2 For…Next

```
<%
Dim i
For i = 1 To 3
    Response.Write "Item " & i & "<br>"
Next
%>
```

25.7.3 Return value

Control statements do not return values. They only control execution order.

25.8 How It Works

- AxonASP parses HTML and server blocks in one page request.
- Server-side VBScript runs first.
- The resulting output is combined with static HTML and sent to the client.

25.9 API Reference

Syntax element	Form	Returns
Script block	`<% statements %>`	No direct inline value
Expression block	`<%= expression %>`	Writes String representation of expression
Output method	`Response.Write expression`	No return value
Directive block	`<%@ ... %>`	No return value
Comment	`' comment`	No return value
Conditional	`If ... Then ... End If`	No return value
Loop	`For ... Next`	No return value

26 Handle ASP Form Data

26.1 Overview

Use `Request.Form` to read fields submitted in an HTTP `POST` request in G3Pix AxonASP. Each key matches an HTML input `name` attribute.

`Request.Form` is built in. Do not instantiate it with `Server.CreateObject`.

26.2 Prerequisites

- The page must receive data using HTTP `POST`.
- The submitted fields must include `name` attributes.
- Do not call `Request.BinaryRead` in the same request where you use `Request.Form`.

26.3 Read a Single Field

26.3.1 Syntax

```
value = Request.Form("fieldName")
```

26.3.2 Return value

Returns a String containing the first submitted value for `fieldName`. Returns an empty String when `fieldName` is not present.

26.3.3 Example

```
<%
Dim userName
userName = Trim(Request.Form("username"))

If userName = "" Then
    Response.Write "Username is required."
    Response.End
End If

Response.Write "Welcome, " & Server.HTMLEncode(userName)
%>
```

26.4 Read Multi-Value Fields

26.4.1 Syntax

```
value = Request.Form("fieldName")(index)
count = Request.Form("fieldName").Count
```

26.4.2 Return value

- `Request.Form("fieldName")(index)` returns a String containing the value at the 1-based `index`.
- `Request.Form("fieldName").Count` returns an Integer with the number of values for that field.

26.4.3 Example

```
<%
Dim i, selectedTagCount
selectedTagCount = Request.Form("tags").Count

For i = 1 To selectedTagCount
    Response.Write "Tag " & i & ": " & Server.HTMLEncode(Request.Form("ta
gs")(i)) & "<br>"
Next
%>
```

26.5 Enumerate Posted Keys

26.5.1 Syntax

```
For Each key In Request.Form
    ' ...
Next
```

26.5.2 Return value

`For Each` returns each key name as a String.

26.5.3 Example

```
<%
Dim key
For Each key In Request.Form
    Response.Write key & " = " & Server.HTMLEncode(Request.Form(key)) & "
<br>"
Next
%>
```

26.6 How It Works

- The form collection is parsed from the request body when first accessed.
- Keys are case-insensitive.
- If you call `Request.BinaryRead`, AxonASP treats the body as raw bytes and does not provide `Request.Form` for that request.

26.7 API Reference

Member	Syntax	Returns
Field value	`Request.Form("fieldName")`	String (first value for field, or empty String if missing)
Indexed value	`Request.Form("fieldName")(index)`	String (value at 1-based index)
Value count	`Request.Form("fieldName").Count`	Integer (number of values for field)
Key enumeration	`For Each key In Request.Form`	String key per iteration

27 Use ASP Cookies

27.1 Overview

Cookies in G3Pix AxonASP are used to store small amounts of data on the client's browser. They are essential for maintaining user state, persisting preferences, and identifying sessions across multiple requests. In AxonASP, cookies are managed through the **Request.Cookies** collection (for reading) and the **Response.Cookies** collection (for writing).

27.2 Prerequisites

- **Client Support**: The client browser must be configured to accept cookies.
- **Header Synchronization**: Cookie modifications via `Response.Cookies` must occur before any HTML or data is flushed to the client, unless buffering is enabled.

27.3 How it Works

Cookies are passed between the server and the client via HTTP headers. - **Reading**: When a browser makes a request, it includes all cookies associated with the domain. AxonASP populates the **Request.Cookies** collection with these values. - **Writing**: When you set a value in **Response.Cookies**, AxonASP includes a `Set-Cookie` header in the HTTP response. - **Subkeys**: A single cookie can store multiple key-value pairs, known as subkeys, which helps organize data without exceeding browser cookie limits.

27.4 API Reference

27.4.1 Request.Cookies

- **Item(name)**: Returns a **String** containing the cookie value.
- **HasKeys**: Returns a **Boolean** indicating if the cookie contains subkeys.

27.4.2 Response.Cookies

- **Domain**: Sets a **String** specifying the domain for which the cookie is valid.
- **Expires**: Sets a **Date** specifying the expiration date and time of the cookie.
- **HttpOnly**: Sets a **Boolean** to prevent client-side script access to the cookie (Security feature).
- **Path**: Sets a **String** specifying the subset of paths on the server for which the cookie is valid.
- **Secure**: Sets a **Boolean** indicating if the cookie should only be transmitted over HTTPS.

27.5 Code Example

The following example demonstrates how to write a persistent cookie with subkeys and then read one of the values back.

```
<%
' Writing a cookie with subkeys and an expiration date
Response.Cookies("UserSettings")("Theme") = "Dark"
Response.Cookies("UserSettings")("FontSize") = "14px"
Response.Cookies("UserSettings").Expires = DateAdd("m", 1, Now())
Response.Cookies("UserSettings").HttpOnly = True

' Reading a cookie value
Dim currentTheme
currentTheme = Request.Cookies("UserSettings")("Theme")

If currentTheme <> "" Then
    Response.Write "Current User Theme: " & currentTheme
End If
%>
```

27.6 Remarks

- **Size Limits**: Browsers typically limit cookies to 4KB each. Use them for identifiers or small settings, not large datasets.
- **Security**: Always set the **HttpOnly** property to **True** for sensitive cookies to mitigate cross-site scripting (XSS) risks.
- **Expiration**: If no expiration date is set, the cookie is treated as a session cookie and will be deleted when the browser is closed.

28 Request Object

28.1 Overview

The `Request` object provides access to all information sent to the server by the current HTTP request. This includes query string parameters, HTML form data, cookies, server environment variables, and client certificate fields. You can read any of these through named collections, and the `Request` object itself supports a default shortcut that searches all collections in sequence.

The `Request` object is always available in every ASP script â€” no `Server.CreateObject` is needed.

28.2 Collections

28.2.1 Request.QueryString

Returns values from the URL query string. For example, the URL `/page.asp?color=blue&size=10` populates `QueryString` with `color` and `size`.

Syntax:

```
Request.QueryString("key")
Request.QueryString("key")(index)   ' 1-based index for multi-value keys
Request.QueryString("key").Count
```

Example:

```
<%
Dim color
color = Request.QueryString("color")
If color = "" Then
    color = "red"
End If
Response.Write "Selected color: " & color
%>
```

```
<%
' Enumerate all query string parameters
Dim k
For Each k In Request.QueryString
    Response.Write k & " = " & Request.QueryString(k) & "<br>"
Next
%>
```

28.2.2 Request.Form

Returns values submitted via an HTML form using the `POST` method. The keys correspond to the `name` attributes of the form elements.

Syntax:

```
Request.Form("fieldname")
Request.Form("fieldname")(index)   ' 1-based, for multi-select
Request.Form("fieldname").Count
```

Note: Accessing `Request.Form` and calling `Request.BinaryRead` in the same script is not allowed. Use one or the other.

Example:

```
<%
Dim username, password
username = Request.Form("username")
password = Request.Form("password")

If Len(Trim(username)) = 0 Then
    Response.Write "Username is required."
    Response.End
End If

Response.Write "Welcome, " & Server.HTMLEncode(username)
%>
```

28.2.3 Request.Cookies

Returns cookie values sent by the browser. Cookies with sub-keys (created using `Response.Cookies("name")("subkey")`) can be read with an extra subscript.

Syntax:

```
Request.Cookies("name")
Request.Cookies("name")("subkey")
Request.Cookies("name").HasKeys        ' True if cookie has sub-keys
```

Example:

```
<%
Dim sessionToken
sessionToken = Request.Cookies("AuthToken")
If sessionToken = "" Then
    Response.Redirect "/login.asp"
End If
%>
```

28.2.4 Request.ServerVariables

Returns the value of HTTP headers and server environment variables. The variable names are standardized strings like `HTTP_HOST`, `REMOTE_ADDR`, `REQUEST_METHOD`, and so on.

Syntax:

```
Request.ServerVariables("variable_name")
```

Common Variables:

Variable	Description
`HTTP_HOST`	The hostname from the request
`HTTP_REFERER`	The referring page URL
`HTTP_USER_AGENT`	Browser/client identification string
`REMOTE_ADDR`	Client IP address
`REQUEST_METHOD`	`GET`, `POST`, `PUT`, etc.
`SCRIPT_NAME`	Path of the current script
`SERVER_NAME`	Server hostname
`SERVER_PORT`	Port number (e.g., `80`)
`SERVER_PROTOCOL`	`HTTP/1.1`
`HTTPS`	`on` if HTTPS, otherwise empty
`QUERY_STRING`	Raw query string
`CONTENT_TYPE`	Content type of POST body
`CONTENT_LENGTH`	Byte count of the POST body

Example:

```
<%
Dim clientIP, method
clientIP = Request.ServerVariables("REMOTE_ADDR")
method   = Request.ServerVariables("REQUEST_METHOD")
```

```
Response.Write "Your IP: " & clientIP & "<br>"
Response.Write "Method:  " & method
%>
```

28.2.5 Request.ClientCertificate

Returns fields from the client's SSL certificate when the connection uses certificate-based authentication.

Syntax:

```
Request.ClientCertificate("key")
```

Common Keys: `Subject`, `Issuer`, `ValidFrom`, `ValidUntil`, `SerialNumber`, `Certificate`

28.2.6 Request (Default Collection)

When a key is passed directly to `Request(key)`, AxonASP searches all collections in this order: 1. `QueryString` 2. `Form` 3. `Cookies` 4. `ClientCertificate` 5. `ServerVariables`

This is a convenience shortcut. For clarity and performance, prefer specifying the collection explicitly.

Example:

```
<%
' Reads from whichever collection contains "id"
Dim id
id = Request("id")
%>
```

28.3 Properties

28.3.1 TotalBytes

Returns the total number of bytes in the request body. This is useful before calling `BinaryRead`.

Syntax:

```
Request.TotalBytes
```

Type: Long (read-only)

Example:

```
<%
Dim bodySize
bodySize = Request.TotalBytes
Response.Write "Body size: " & bodySize & " bytes"
%>
```

28.4 Methods

28.4.1 BinaryRead

Reads the raw bytes from the POST body. This is used when you need to process binary data â€” for example, a file upload or raw JSON body not processed through `Request.Form`.

After calling `BinaryRead`, accessing `Request.Form` is not permitted in the same request.

Syntax:

```
Request.BinaryRead(count)
```

Parameters:

Parameter	Type	Description
count	Long	Number of bytes to read

Return Value: Variant (Byte Array)

Example:

```
<%
Dim rawBytes, byteCount
byteCount = Request.TotalBytes
rawBytes  = Request.BinaryRead(byteCount)
' rawBytes is now a byte safearray containing the POST body
%>
```

28.5 Remarks

- All collection lookups are **case-insensitive**.
- Multi-value form fields and query string parameters are accessible by 1-based numeric index.

- Iterating `For Each k In Request.QueryString` yields collection keys in insertion order.
- `Request.Form` is lazily loaded on first access; calling `BinaryRead` before it prevents form parsing.
- The default collection lookup order (`QueryString â†’ Form â†’ Cookies â†’ ClientCertificate â†’ ServerVariables`) applies only when you omit the collection name.

29 Response Object

29.1 Overview

The `Response` object controls everything that AxonASP sends back to the client browser. Through it, you write HTML output, set HTTP headers and cookies, control caching, issue redirects, and manage response buffering. It is available in every ASP script with no setup required.

29.2 Methods

29.2.1 Response.Write

Writes a string directly to the HTTP response. This is the primary method for generating output in an ASP page. The `<%= expression %>` shortcut is equivalent to `Response.Write expression`.

Syntax:

```
Response.Write(string)
<%= expression %>
```

Parameters:

Parameter	Type	Description
string	String/Variant	The content to output

Return Value: None

Example:

```
<%
Response.Write "<h1>Hello, World!</h1>"
Response.Write "Today is: " & Date()
%>
<p>Current time: <%= Time() %></p>
```

29.2.2 Response.BinaryWrite

Writes raw binary data to the response. Used when sending file downloads, images, or other non-text content directly from a byte array.

Syntax:

```
Response.BinaryWrite(data)
```

Parameters:

Parameter	Type	Description
data	Variant (Byte Array)	Binary content to write

Return Value: None

Example:

```
<%
Response.ContentType = "image/png"
Response.BinaryWrite imageByteArray
%>
```

29.2.3 Response.Redirect

Issues an HTTP 302 redirect to the specified URL and immediately stops the current script. Any buffered output is discarded.

Syntax:

```
Response.Redirect(url)
```

Parameters:

Parameter	Type	Description
url	String	Absolute or relative URL to redirect to

Return Value: None (does not return â€” execution stops)

Example:

```
<%
If Session("UserID") = "" Then
    Response.Redirect "/login.asp?reason=notloggedin"
End If
' Execution continues only when user is logged in
%>
```

29.2.4 Response.End

Immediately stops script execution and sends the current buffer to the client. Any code after `Response.End` does not run.

Syntax:

```
Response.End()
```

Return Value: None (does not return)

Example:

```
<%
If Request.QueryString("debug") = "1" Then
    Response.Write "Debug mode active"
    Response.End
End If
' Normal page code continues here
%>
```

29.2.5 Response.Flush

Forces any currently buffered output to be sent to the client immediately. Useful for long-running operations where you want to show progress.

Syntax:

```
Response.Flush()
```

Return Value: None

Example:

```
<%
Response.Write "Starting process...<br>"
Response.Flush

' Simulate work
Dim i
For i = 1 To 5
    Response.Write "Step " & i & " complete<br>"
    Response.Flush
Next

Response.Write "Done."
%>
```

29.2.6 Response.Clear

Discards all content currently in the output buffer. Only works when buffering is enabled. Has no effect after `Response.Flush` has been called.

Syntax:

```
Response.Clear()
```

Return Value: None

Example:

```
<%
On Error Resume Next
' Attempt to build output
Response.Write BuildPage()
If Err.Number <> 0 Then
    Response.Clear
    Response.Write "<p>An error occurred. Please try again.</p>"
End If
%>
```

29.2.7 Response.AddHeader

Adds or replaces an HTTP response header. Headers must be set before any output is flushed to the client.

Syntax:

```
Response.AddHeader(name, value)
```

Parameters:

Parameter	Type	Description
name	String	HTTP header name
value	String	Header value

Return Value: None

Example:

```
<%
Response.AddHeader "X-Content-Type-Options", "nosniff"
Response.AddHeader "X-Frame-Options", "DENY"
Response.AddHeader "Content-Disposition", "attachment; filename=""report.
```

```
csv"""
%>
```

29.2.8 Response.AppendToLog

Appends a custom message to the AxonASP server log file (`temp/server.log`). Useful for application-level audit trails without file system access.

Syntax:

```
Response.AppendToLog(message)
```

Parameters:

Parameter	Type	Description
message	String	Message to write to the log

Return Value: None

Example:

```
<%
Response.AppendToLog "User " & Session("UserID") & " accessed report at "
 & Now()
%>
```

29.3 Properties

29.3.1 Response.Buffer

Controls whether output is buffered before being sent to the client. When `True` (the default), all output is held in memory until the script ends or `Flush` is called. When `False`, each `Response.Write` is sent immediately.

Type: Boolean (read/write)
Default: `True`

Example:

```
<%
Response.Buffer = False   ' Disable buffering â€” immediate output
Response.Write "This line goes to the browser right away."
%>
```

29.3.2 Response.ContentType

Sets the MIME type of the response. The default is `text/html`. Change this when serving JSON, XML, CSV, images, or other content types.

Type: String (read/write)
Default: `"text/html"`

Example:

```
<%
Response.ContentType = "application/json"
Response.Write "{""status"": ""ok""}"
%>
```

29.3.3 Response.Charset

Sets the character encoding appended to the `Content-Type` header. The value must be a valid IANA charset name.

Type: String (read/write)
Default: `"utf-8"`

Example:

```
<%
Response.Charset = "utf-8"
%>
```

29.3.4 Response.Status

Sets the HTTP status line. Changing this before any output is flushed overrides the default `200 OK`.

Type: String (read/write)
Default: `"200 OK"`

Example:

```
<%
Response.Status = "404 Not Found"
Response.Write "<h1>Page not found</h1>"
%>

<%
Response.Status = "403 Forbidden"
```

```
Response.End
%>
```

29.3.5 Response.CacheControl

Sets the value of the `Cache-Control` HTTP header. Use `"Public"` to allow proxies to cache the response or `"Private"` (the default) to restrict caching to the client only.

Type: String (read/write)
Default: `"Private"`

Example:

```
<%
Response.CacheControl = "Public"
Response.Expires = 60    ' cache for 60 minutes
%>
```

29.3.6 Response.Expires

Sets the number of minutes from the current time before the response expires. A value of `0` or negative means the content is already expired (forces revalidation).

Type: Integer (read/write, minutes)

Example:

```
<%
Response.Expires = 0      ' Force browsers to revalidate
Response.CacheControl = "no-cache"
%>
```

29.3.7 Response.ExpiresAbsolute

Sets an absolute date and time for the response to expire, rather than a relative offset.

Type: Date/String (read/write)

Example:

```
<%
Response.ExpiresAbsolute = "April 30, 2026 00:00:00"
%>
```

29.3.8 Response.Cookies

Provides access to the collection of response cookies. Each entry can be a simple string value or a collection of sub-key/value pairs. Cookie attributes (domain, path, expiration, secure, HttpOnly) are set as properties of each cookie entry.

Syntax:

```
Response.Cookies("name") = "value"
Response.Cookies("name").Expires = DateAdd("d", 30, Now())
Response.Cookies("name").Domain  = "example.com"
Response.Cookies("name").Path    = "/"
Response.Cookies("name").Secure  = True
Response.Cookies("name").HTTPOnly = True

' Dictionary-style cookie (sub-keys)
Response.Cookies("prefs")("theme") = "dark"
Response.Cookies("prefs")("lang")  = "en"
Response.Cookies("prefs").Expires  = DateAdd("d", 365, Now())
```

Example:

```
<%
Response.Cookies("SessionToken") = "abc123xyz"
Response.Cookies("SessionToken").Path    = "/"
Response.Cookies("SessionToken").Secure  = True
Response.Cookies("SessionToken").HTTPOnly = True
Response.Cookies("SessionToken").Expires = DateAdd("d", 1, Now())
%>
```

29.3.9 Response.IsClientConnected

Returns `True` if the client is still connected and receiving the response. Useful to abort long-running scripts when the browser has disconnected.

Type: Boolean (read-only)

Example:

```
<%
Dim i
For i = 1 To 1000000
    If Not Response.IsClientConnected Then
        Response.End
    End If
    ' ... process record i ...
```

```
Next
%>
```

29.3.10 Response.CodePage

Sets the code page used to convert Unicode strings to bytes in the response. `65001` is UTF-8.

Type: Integer (read/write)
Default: `65001` (UTF-8)

29.3.11 Response.PICS

Sets the value of a `PICS-Label` HTTP header for content rating systems.

Type: String (read/write)

29.4 Remarks

- **Buffering** is enabled by default. The buffer has a configurable maximum size (`response_buffer_limit_mb` in `axonasp.toml`). Exceeding it raises a runtime error.
- Calling `Response.End` or `Response.Redirect` causes a non-resumable signal that terminates script execution immediately â€” code after them in the same script does not run.
- Setting headers or cookies after `Response.Flush` has been called has no effect; headers must be sent before any body bytes.
- `Response.Write` accepts any VBScript Variant; the value is coerced to a string before output.

30 Server Object

30.1 Overview

The `Server` object provides utility methods used throughout ASP scripts. Its most common uses are creating COM/native objects, translating virtual paths to physical file paths, and encoding strings for safe output or URL use. It is available in every ASP script with no setup.

30.2 Methods

30.2.1 Server.CreateObject

Instantiates a COM object or one of AxonASP's native library objects by its ProgID string. This is the main entry point for all built-in libraries such as G3JSON, G3DB, G3Mail, ADODB, Scripting.FileSystemObject, and so on.

Syntax:

```
Set obj = Server.CreateObject("ProgID")
```

Parameters:

Parameter	Type	Description
ProgID	String	The programmatic identifier of the object to create

Return Value: Object (use `Set` assignment)

Example:

```
<%
Set json = Server.CreateObject("G3JSON")
Dim data : data = json.Parse("{""name"":""Alice"",""age"":30}")
Response.Write data("name")    ' Alice
Set json = Nothing
%>

<%
Set conn = Server.CreateObject("ADODB.Connection")
conn.Open "Provider=...; Data Source=..."
' ...
conn.Close
Set conn = Nothing
%>
```

30.2.2 Server.MapPath

Translates a virtual (URL-style) path into an absolute physical file system path on the server. This is needed whenever you must open a file using `FileSystemObject` or another file-access library.

Syntax:

```
Server.MapPath(path)
```

Parameters:

Parameter	Type	Description
path	String	Virtual path. A leading / means the web root. A relative path is relative to the current script's directory.

Return Value: String (absolute file system path)

Example:

```
<%
' Absolute virtual path
Dim fullPath
fullPath = Server.MapPath("/data/config.json")
Response.Write fullPath    ' e.g., C:\inetpub\wwwroot\data\config.json

' Relative path (relative to current script's directory)
Dim relPath
relPath = Server.MapPath("images/logo.png")
Response.Write relPath
%>
```

30.2.3 Server.HTMLEncode

Escapes HTML special characters so that a string is safe to insert into an HTML page. Converts <, >, &, ", and ' to their HTML entity equivalents.

Syntax:

```
Server.HTMLEncode(string)
```

Parameters:

Parameter	Type	Description
string	String	The text to escape

Return Value: String (HTML-safe text)

Example:

```
<%
Dim userInput : userInput = Request.Form("comment")
Response.Write "<p>" & Server.HTMLEncode(userInput) & "</p>"
' Prevents XSS: <script>alert(1)</script> becomes
' &lt;script&gt;alert(1)&lt;/script&gt;
%>
```

30.2.4 Server.URLEncode

Encodes a string for safe use in a URL query string. Spaces become +, and special characters are percent-encoded.

Syntax:

```
Server.URLEncode(string)
```

Parameters:

Parameter	Type	Description
string	String	The value to encode

Return Value: String (URL-encoded)

Example:

```
<%
Dim searchTerm : searchTerm = "hello world & more"
Response.Redirect "/search.asp?q=" & Server.URLEncode(searchTerm)
' Redirects to /search.asp?q=hello+world+%26+more
%>
```

30.2.5 Server.URLPathEncode

Encodes a string for use in a URL path segment. Slashes (/) are preserved as-is; only individual path components are encoded.

Syntax:

```
Server.URLPathEncode(string)
```

Parameters:

Parameter	Type	Description
string	String	The path to encode

Return Value: String (path-encoded)

Example:

```
<%
Dim folder : folder = "my documents/2026"
Dim encodedPath : encodedPath = Server.URLPathEncode(folder)
' Result: "my%20documents/2026"  (slash preserved)
%>
```

30.2.6 Server.GetLastError

Returns the `ASPError` object populated with information about the most recent error in the current script. Useful inside custom error handlers.

Syntax:

```
Set err = Server.GetLastError()
```

Return Value: ASPError object

Example:

```
<%
On Error Resume Next
Dim x : x = 1 / 0   ' Division by zero
If Err.Number <> 0 Then
    Dim aspErr
    Set aspErr = Server.GetLastError()
    Response.Write "Error " & aspErr.Number & ": " & aspErr.Description
End If
%>
```

30.2.7 Server.Execute

Executes another ASP file as if its content were inline in the current script. The called script shares the same `Request`, `Response`, `Session`, and `Application` objects. Execution returns to the calling page when the included script finishes.

Syntax:

```
Server.Execute(path)
```

Parameters:

Parameter	Type	Description
path	String	Virtual path to the ASP file to execute

Return Value: None

Example:

```
<%
' include a header component
Server.Execute "/includes/header.asp"
%>
<main>Page content here</main>
```

```
<%
Server.Execute "/includes/footer.asp"
%>
```

30.2.8 Server.Transfer

Transfers control to another ASP file without issuing an HTTP redirect. The URL in the browser does not change. Unlike `Execute`, control does not return to the calling page.

Syntax:

```
Server.Transfer(path)
```

Parameters:

Parameter	Type	Description
path	String	Virtual path to the ASP file

Return Value: None (execution continues in the target file)

Example:

```
<%
If Session("UserID") = "" Then
    Server.Transfer "/errors/unauthorized.asp"
End If
%>
```

30.3 Properties

30.3.1 Server.ScriptTimeout

Gets or sets the maximum number of seconds that a script is allowed to run before the server terminates it with a timeout error. The default is set by `default_script_timeout` in `axonasp.toml` (default 60 seconds).

Type: Integer (read/write, seconds)

Example:

```
<%
' Allow a long-running export up to 5 minutes
Server.ScriptTimeout = 300

' ... export logic ...
%>
```

30.4 Remarks

- `Server.MapPath` is sandboxed to the web root configured in `axonasp.toml`. Paths outside the web root resolve to the web root itself.
- `Server.HTMLEncode` should be applied to **all user-supplied content** before writing it to HTML output to prevent cross-site scripting (XSS).
- `Server.CreateObject` accepts the same ProgIDs as classic IIS ASP plus all AxonASP-specific libraries. A full list is in the Libraries section.
- `Server.Execute` and `Server.Transfer` can be used as alternatives to `#include` for dynamic component inclusion.

31 Manage Session State

31.1 Overview

Use the `Session` object to persist user-specific values between HTTP requests in G3Pix AxonASP. The runtime identifies each visitor with an `ASPSESSIONID` cookie and maps that cookie to a server-side session store.

`Session` is built in. Do not instantiate it with `Server.CreateObject`.

31.2 Prerequisites

- The client must accept cookies.
- Session storage must be enabled in your AxonASP runtime configuration.

31.3 Store and Read Values

Use dictionary-style access to save and read values by key.

```
Session("UserID") = 42
Session("UserName") = "alice"

Dim currentUserName
currentUserName = Session("UserName")
```

31.3.1 Return details for `Session("key")`

- Returns `Empty` when the key does not exist.
- Returns the exact value previously stored for that key when it exists.

31.4 Methods

31.4.1 Session.Abandon

Ends the current session and clears all values in that session.

31.4.1.1 Syntax

```
Session.Abandon
```

31.4.1.2 Return value

Returns no value.

31.4.1.3 Example

```
<%
Session.Abandon
Response.Redirect "/login.asp?msg=loggedout"
%>
```

31.5 Properties

31.5.1 Session.SessionID

Gets the current session identifier.

31.5.1.1 Type

String (read-only)

31.5.1.2 Return value

Returns a non-empty String that identifies the current session.

31.5.1.3 Example

```
<%
Response.Write "Session ID: " & Session.SessionID
%>
```

31.5.2 Session.Timeout

Gets or sets the inactivity timeout in minutes.

31.5.2.1 Type

Integer (read/write)

31.5.2.2 Return value

Returns an Integer with the configured inactivity timeout.

31.5.2.3 Example

```
<%
If Session("Role") = "premium" Then
    Session.Timeout = 120
End If
%>
```

31.5.3 Session.LCID

Gets or sets the locale identifier used by VBScript formatting functions in the current session.

31.5.3.1 Type

Integer (read/write)

31.5.3.2 Return value

Returns an Integer LCID value such as `1033`.

31.5.3.3 Example

```
<%
Session.LCID = 1033
Response.Write FormatDateTime(Now(), 1)
%>
```

31.5.4 Session.CodePage

Gets or sets the session code page.

31.5.4.1 Type

Integer (read/write)

31.5.4.2 Return value

Returns an Integer code page value such as `65001`.

31.5.4.3 Example

```
<%
Session.CodePage = 65001
%>
```

31.5.5 Session.Contents

Accesses all key/value pairs stored in the current session.

31.5.5.1 Syntax

```
Session.Contents("key")
Session.Contents.Remove "key"
Session.Contents.RemoveAll
```

31.5.5.2 Return value

- `Session.Contents("key")` returns `Empty` when the key does not exist.
- `Session.Contents("key")` returns the stored value when the key exists.
- `Session.Contents.Remove "key"` returns no value.

- `Session.Contents.RemoveAll` returns no value.

31.5.5.3 Example

```
<%
Dim k
For Each k In Session.Contents
    Response.Write k & " = " & CStr(Session.Contents(k)) & "<br>"
Next

Session.Contents.Remove "CartItems"
%>
```

31.5.6 Session.StaticObjects

Gets the collection of session-scoped objects declared in `global.asa` with `SCOPE="Session"`.

31.5.6.1 Type

Collection (read-only)

31.5.6.2 Return value

Returns a collection object. It can be enumerated but not replaced.

31.6 How It Works

- G3Pix AxonASP associates one `ASPSESSIONID` cookie with one server-side session record.
- Session keys are case-insensitive.
- When you call `Session.Abandon`, the current record is discarded. A later request can create a new session with a different ID.

31.7 Best Practices

- Store compact values such as IDs and flags.
- Avoid large arrays and large object graphs in session state.
- Set `Session.Timeout` explicitly for workflows that require longer inactivity windows.

32 Use the ASP Application Object

32.1 Overview

The **Application** object in G3Pix AxonASP stores data that is shared across all users and all requests for the entire lifetime of the web application. Unlike the **Session** object, which is unique to each visitor, **Application** values are global. This object is

typically used for site-wide settings, application-level counters, or caching shared lookup data.

32.2 Prerequisites

- **Global.asa**: Initial application values are often set in the `Application_OnStart` event.
- **Concurrency Awareness**: Because multiple requests access this object simultaneously, you must use locking methods to prevent data collisions.

32.3 How it Works

The **Application** object acts as a global dictionary. When you store a value, it remains in memory until the AxonASP server process is restarted. - **Locking**: Use the **Lock** method before modifying any value to ensure thread safety. - **Unlocking**: Always call the **Unlock** method immediately after your modification is complete. - **Persistence**: Data is lost if the server shuts down or the application pool is recycled.

32.4 API Reference

32.4.1 Methods

- **Lock**: Prevents other clients from modifying the **Application** object. Returns **Empty**.
- **Unlock**: Releases the lock on the **Application** object, allowing other clients to modify it. Returns **Empty**.

32.4.2 Collections

- **Contents**: Returns a collection of all values added to the application through script commands.
- **StaticObjects**: Returns a collection of all objects added to the application with the `<OBJECT>` tag in `global.asa`.

32.5 Code Example

The following example demonstrates how to safely increment a global page hit counter using the **Lock** and **Unlock** methods.

```
<%
' Access the Application object
Application.Lock
Application("TotalHits") = Application("TotalHits") + 1
Application.Unlock

Response.Write "Total application views: " & Application("TotalHits")
%>
```

32.6 Remarks

- **Thread Safety**: Failing to use **Lock** and **Unlock** when writing to the **Application** object can result in lost updates in high-concurrency environments.
- **Memory Management**: Avoid storing large objects or frequently changing data in the **Application** object to prevent memory bloat.
- **Redirection**: Never call `Response.Redirect` or `Response.End` while an application lock is active, as this may bypass the **Unlock** call and freeze the object for other threads.

33 Use the ASPError Object

33.1 Overview

The **ASPError** object in G3Pix AxonASP provides structured, detailed information about the most recent error that occurred during script execution. It is primarily used within custom error-handling pages to display or log technical data such as the error number, source file, and line number where the fault occurred.

33.2 Prerequisites

- **Server.GetLastError**: You must call the `Server.GetLastError` method to retrieve a reference to the **ASPError** object.
- **Custom Error Pages**: To use this object effectively, configure your application to redirect to a custom error script (e.g., via `web.config`).

33.3 How it Works

When a runtime or compilation error occurs in G3Pix AxonASP, the engine captures the state of the error. This information is preserved until it is explicitly retrieved. - **Read-Only**: All properties of the **ASPError** object are read-only. - **Scope**: The object only contains information about the *last* error that occurred in the current request context.

33.4 API Reference

33.4.1 Properties

- **ASPCode**: Returns an **Integer** representing the error code generated by the AxonASP engine.
- **ASPDescription**: Returns a **String** containing a detailed description of the error if it is an ASP-specific error.
- **Category**: Returns a **String** indicating the source of the error (e.g., “VBScript runtime”, “AxonASP”, or “ASP”).
- **Column**: Returns an **Integer** specifying the character position within the line where the error occurred.

- **Description**: Returns a **String** containing a short description of the error.
- **File**: Returns a **String** containing the virtual path of the file that generated the error.
- **HelpContext**: Returns an **Integer** specifying the help context ID for the error (usually 0).
- **HelpFile**: Returns a **String** containing the path to a help file if available.
- **Line**: Returns an **Integer** specifying the line number in the source file where the error occurred.
- **Number**: Returns an **Integer** representing the standard VBScript error number.
- **Source**: Returns a **String** containing the source code of the line that caused the error.

33.5 Code Example

The following example demonstrates how to extract error details in a custom error handler page.

```
<%
Dim objErr
Set objErr = Server.GetLastError()

Response.Write "<h1>An Error Occurred</h1>"
Response.Write "<b>Category:</b> " & objErr.Category & "<br>"
Response.Write "<b>Number:</b> " & objErr.Number & "<br>"
Response.Write "<b>Description:</b> " & objErr.Description & "<br>"
Response.Write "<b>File:</b> " & objErr.File & "<br>"
Response.Write "<b>Line:</b> " & objErr.Line & "<br>"

Set objErr = Nothing
%>
```

33.6 Remarks

- **Security**: In production environments, avoid displaying the full details of the **ASPError** object to end-users, as it may reveal sensitive system information.
- **Logging**: Use the properties of this object to populate database logs or system event logs for administrative review.
- **Persistence**: The error information is only available for the duration of the request that triggered the error or the subsequent error handler.

34 Use the ASP Dictionary Object

34.1 Overview

The **Dictionary** object in G3Pix AxonASP is a highly efficient data structure used to store and manage key-value pairs. It functions similarly to an associative array or a

hash map, allowing for fast retrieval of data based on unique keys. This object is preferred over standard arrays for tasks involving lookup tables, JSON-like maps, and managing complex configuration data.

34.2 Syntax

To instantiate the object, use the following syntax:

```
Set dict = Server.CreateObject("Scripting.Dictionary")
```

34.3 How it Works

The AxonASP Virtual Machine provides a native implementation of the **Scripting.Dictionary** object. - **Keys and Items**: Each entry consists of a unique **Key** and an associated **Item** (value). - **Type Flexibility**: Keys and items can be any Variant type, including strings, numbers, and other object references. - **Comparison Mode**: The object can be configured to perform case-sensitive or case-insensitive key comparisons.

34.4 API Reference

34.4.1 Methods

- **Add(key, item)**: Adds a new key-value pair to the dictionary. Returns **Empty**.
- **Exists(key)**: Returns a **Boolean** indicating whether the specified key exists in the dictionary.
- **Items()**: Returns a **VBArray** containing all the items in the dictionary.
- **Keys()**: Returns a **VBArray** containing all the keys in the dictionary.
- **Remove(key)**: Removes a specific key-value pair from the dictionary. Returns **Empty**.
- **RemoveAll()**: Clears all entries from the dictionary. Returns **Empty**.

34.4.2 Properties

- **CompareMode**: Gets or sets an **Integer** specifying the comparison mode (0 for Binary/Case-Sensitive, 1 for Text/Case-Insensitive).
- **Count**: Returns an **Integer** representing the number of entries currently in the dictionary.
- **Item(key)**: Gets or sets the value for a specified key. This is the default property.
- **Key(oldKey)**: Sets a new name for an existing key.

34.5 Code Example

The following example demonstrates how to create a dictionary, add data, and iterate through the keys.

```asp
<%
Dim dict, k
Set dict = Server.CreateObject("Scripting.Dictionary")

' Set case-insensitive mode
dict.CompareMode = 1

' Populate data
dict.Add "Engine", "AxonASP"
dict.Add "Version", "2.0"
dict.Add "Status", "Running"

' Check for existence and retrieve item
If dict.Exists("Engine") Then
    Response.Write "Application Engine: " & dict("Engine") & "<br>"
End If

' Iterate through all entries
For Each k In dict.Keys
    Response.Write k & ": " & dict(k) & "<br>"
Next

Set dict = Nothing
%>
```

34.6 Remarks

- **Unique Keys**: Attempting to add a key that already exists will trigger a runtime error. Use the **Exists** method to check before adding.
- **Resource Management**: Always set the dictionary object to **Nothing** once it is no longer needed to free system memory.
- **Performance**: The **Dictionary** object is optimized for fast lookups even with large datasets.

35 Use G3Pix AxonASP Library Objects

35.1 Overview

Use this page as a quick reference for object creation through `Server.CreateObject` in G3Pix AxonASP. It summarizes supported primary ProgIDs and common compatibility aliases.

35.2 Prerequisites

- Run code inside a Classic ASP page hosted by G3Pix AxonASP.
- Use `Server.CreateObject("ProgID")` with a supported ProgID string.

35.3 Syntax

```
Set obj = Server.CreateObject("ProgID")
```

35.4 Object Creation Summary

Category	Primary ProgID	Compatibility Aliases
Core functions	G3AXON.FUNCTIONS	G3AXON
Markdown	G3MD	None
Crypto	G3CRYPTO	None
JSON	G3JSON	None
Database helper	G3DB	None
HTTP helper	G3HTTP	None
Mail	G3MAIL	CDONTS.NewMail, CDO.Message, Persits.MailSender
Image	G3IMAGE	None
File helper	G3FILES	None
Template	G3TEMPLATE	None
Archive and compression	G3ZIP, G3ZLIB, G3TAR, G3ZSTD, G3FC	None
PDF	G3PDF	None
Upload	G3FILEUPLOADER	None
Shell	WSCRIPT.SHELL	Shell
ADOX	ADOX.CATALOG	None
ADODB	ADODB.CONNECTION, ADODB.RECORDSET, ADODB.COMMAND, ADODB.STREAM, ADODBOLE.CONNECTION	None
Scripting runtime	SCRIPTING.FILESYSTEMOBJECT, SCRIPTING.DICTIONARY	None
XML HTTP	MSXML2.SERVERXMLHTTP	MSXML2.XMLHTTP, MICROSOFT.XMLHTTP
XML DOM	MSXML2.DOMDOCUMENT	MICROSOFT.XMLDOM
MSWC compatibility	MSWC.ADROTATOR, MSWC.BROWSERTYPE, MSWC.NEXTLINK,	None

Category	Primary ProgID	Compatibility Aliases
	MSWC.CONTENTROTATOR, MSWC.COUNTERS, MSWC.PAGECOUNTER, MSWC.TOOLS, MSWC.MYINFO, MSWC.PERMISSIONCHECKER	
RegExp	VBSCRIPT.REGEXP	REGEXP

35.5 Return Value

`Server.CreateObject` returns an object handle bound to the native G3Pix AxonASP implementation for the resolved ProgID.

35.6 Remarks

- **Object creation is case-insensitive for supported ProgIDs.**
- **Unsupported ProgIDs fall through to host-level CreateObject handling.**
- **Compatibility aliases are mapped to native AxonASP implementations where available.**

35.7 Code Example

```
<%
Option Explicit
Dim ax, db, stm, xmlHttp

Set ax = Server.CreateObject("G3AXON.FUNCTIONS")
Set db = Server.CreateObject("ADODB.Connection")
Set stm = Server.CreateObject("ADODB.Stream")
Set xmlHttp = Server.CreateObject("MSXML2.ServerXMLHTTP")

Response.Write TypeName(ax)

Set xmlHttp = Nothing
Set stm = Nothing
Set db = Nothing
Set ax = Nothing
%>
```

35.8 API Reference

- Entry point: `Server.CreateObject`
- Input: ProgID string
- Resolution: native AxonASP ProgID map, then compatibility alias map, then host-level fallback
- Output: object handle for the resolved implementation

36 Use the G3AXON.FUNCTIONS Library

36.1 Overview

The **G3AXON.FUNCTIONS** library provides a comprehensive set of native utility functions for G3Pix AxonASP applications. These functions extend the standard VBScript capabilities with high-performance system operations, environment management, advanced math routines, and string manipulation helpers. The library is optimized for zero-allocation performance and provides direct access to the underlying Go runtime and host operating system features.

36.2 Syntax

To instantiate the library, use the following syntax:

```
Set ax = Server.CreateObject("G3AXON.FUNCTIONS")
```

36.3 Prerequisites

No external dependencies are required. The G3AXON.FUNCTIONS library is a built-in native component of the AxonASP environment.

36.4 How it Works

The G3AXON.FUNCTIONS object acts as a gateway to specialized system and utility routines. - **System & Environment**: Provides access to process IDs, hostname, environment variables, and directory management. - **Math & Numeric**: Includes advanced functions like **AxMax**, **AxMin**, and high-precision random number generation. - **Array & String**: Offers robust tools for data manipulation, such as **AxExplode**, **AxImplode**, and **AxPad**. - **Output**: The **Axw** method provides an HTML-escaped alternative to standard output routines.

Most functions in this library are also available as global built-ins (prefixed with “Ax”) for developer convenience.

36.5 API Reference

This library contains over 80 specialized methods. For detailed information on specific members, please refer to the individual method documentation pages.

36.6 Code Example

The following example demonstrates how to retrieve system information and format a number using the library.

```
<%
Dim ax, sysInfo, formatted
```

```
Set ax = Server.CreateObject("G3AXON.FUNCTIONS")

' Retrieve a summary of system information
sysInfo = ax.AxSystemInfo("a")
Response.Write "System: " & sysInfo & "<br>"

' Format a numeric value with thousands separators
formatted = ax.AxNumberFormat(1234567.89, 2, ".", ",")
Response.Write "Formatted: " & formatted

Set ax = Nothing
%>
```

37 G3AXON.FUNCTIONS Method Reference

37.1 Overview

This page lists every method exposed by the `G3AXON.FUNCTIONS` library. Methods are grouped by functional category. All method names are case-insensitive.

37.2 System and Environment

Method	Returns	Description
`AxCacheDirPath()`	String	Full path to the `.temp/cache/` directory with trailing separator.
`AxChangeDir(path)`	Boolean	Changes the process working directory. Returns `True` on success.
`AxChangeMode(path, mode)`	Boolean	Changes the file mode using an octal string (e.g., `"0644"`).
`AxChangeOwner(path, uid, gid)`	Boolean	Changes file owner and group IDs.
`AxChangeTimes(path, accessTime, modifyTime)`	Boolean	Sets file access and modification timestamps from Unix seconds.
`AxClearEnvironment()`	Boolean	Clears all process environment variables. Always returns `True`.

Method	Returns	Description
`AxCreateLink(sourcePath, linkPath)`	Boolean	Creates a hard link from source to destination.
`AxCurrentDir()`	String	Current process working directory path.
`AxCurrentUser()`	String	Current operating system user name.
`AxDirSeparator()`	String	OS directory separator (\ on Windows, / on Unix).
`AxEffectiveUserId()`	Integer	Effective user ID on Unix, or `-1` on Windows.
`AxEngineName()`	String	AxonASP engine name string.
`AxEnvironmentList()`	Array	All environment entries as `KEY=VALUE` strings.
`AxEnvironmentValue(name[, default])`	String	Value of an environment variable, with optional default fallback.
`AxExecutablePath()`	String	Absolute path of the running AxonASP executable.
`AxExecute(command)`	String/Boolean	Runs a shell command; returns combined stdout/stderr or `False` on empty command.
`AxGetEnv(name)`	String	Value of an environment variable by name.
`AxHostnameValue()`	String	Current machine host name.
`AxIsPathSeparator(char)`	Boolean	`True` if the single character is a valid path separator on the current platform.
`AxPathListSeparator()`	String	OS path-list separator (; on Windows, : on Unix).
`AxProcessId()`	Integer	Current process ID (PID).
`AxRuntimeInfo()`	String	Multi-section plain-text diagnostic report.

Method	Returns	Description
`AxShutdownAxonASPServer()`	Boolean	Terminates the server process if shutdown is enabled in configuration.
`AxSystemInfo([mode])`	String	System information: OS, hostname, architecture, Go version, or all combined.
`AxUserConfigDirPath()`	String	Resolved path to `config/axonasp.toml`.
`AxUserHomeDirPath()`	String	Current user home directory path.
`AxVersion()`	String	AxonASP runtime version string.

37.3 Configuration

Method	Returns	Description
`AxGetConfig(key)`	String	Reads a configuration key from `axonasp.toml` with optional environment override.
`AxGetConfigKeys()`	Array	All available configuration key names.
`AxGetDefaultCss()`	String	Raw content of the CSS file configured at `axfunctions.ax_default_css_path`.
`AxGetLogo()`	String	Configured logo as a Base64 data URI.

37.4 Math and Numeric

Method	Returns	Description
`AxCeil(number)`	Double	Rounds a value up to the nearest integer boundary.
`AxCount(value)`	Integer	Number of elements in a VBScript array. Returns `0`

Method	Returns	Description
		for non-arrays.
`AxFloor(number)`	Double	Rounds a value down to the nearest integer boundary.
`AxFloatPrecisionDigits()`	Integer	Standard float precision digit count (15).
`AxIntegerMax()`	Integer	Maximum native 64-bit integer value (9,223,372,036,854,775,807).
`AxIntegerMin()`	Integer	Minimum native 64-bit integer value (-9,223,372,036,854,775,808).
`AxIntegerSizeBytes()`	Integer	Native integer size in bytes (4 on 32-bit, 8 on 64-bit).
`AxMax(n1, n2, ...)`	Double	Largest value from the supplied arguments.
`AxMin(n1, n2, ...)`	Double	Smallest value from the supplied arguments.
`AxNumberFormat(n, decimals, decPoint, thousandsSep)`	String	Formats a number with configurable separators.
`AxPi()`	Double	Mathematical constant π (3.141592653589793).
`AxPlatformBits()`	Integer	Native integer size in bits (32 or 64).
`AxRand([max] / [min, max])`	Integer	Random integer within the specified range.
`AxSmallestFloatValue()`	Double	Smallest non-zero positive `float64` value (5e-324).

37.5 Array

Method	Returns	Description
`AxArrayReverse(arr)`	Array	New array with elements in reverse order.

Method	Returns	Description
`AxExplode(delimiter, str [, limit])`	Array	Splits a string into an array by delimiter.
`AxImplode(glue, arr)`	String	Joins array elements into a string with a separator.
`AxRange(start, end [, step])`	Array	Numeric array from `start` to `end` with optional `step`.

37.6 String

Method	Returns	Description
`AxNl2Br(str)`	String	Replaces line breaks (CRLF, LF, CR) with HTML ` ` tags.
`AxPad(str, length, padStr, padType)`	String	Pads a string to a target length on the left, right, or both sides.
`AxRepeat(str, count)`	String	Repeats a string a specified number of times.
`AxStringGetCsv(str [, delimiter])`	Array	Parses one CSV row and returns the field values as an array.
`AxStringReplace(search, replacement, subject)`	String	Replaces all occurrences of a substring.
`AxTrim(str [, chars])`	String	Trims whitespace or a custom character set from both ends.
`AxUcFirst(str)`	String	Uppercases the first character of a string.
`AxW(text)`	Empty	Writes HTML-escaped text to the response output stream.
`AxWordCount(str [, format])`	Integer/Array	Counts words in a string, or returns them as an array.

37.7 Hash and Encoding

Method	Returns	Description
`AxBase64Decode(str)`	String	Decodes a Base64 string. Returns empty string on invalid input.
`AxBase64Encode(str)`	String	Encodes a string to Base64.
`AxHash(algo, str)`	String	Hexadecimal hash using `"md5"`, `"sha1"`, or `"sha256"`.
`AxHexToRgb(hex)`	String	Converts `#RRGGBB` or `#RGB` to `rgb(R,G,B)`.
`AxHtmlSpecialChars(str)`	String	Escapes `&`, `<`, `>`, and `"` to HTML entities.
`AxMD5(str)`	String	32-character lowercase MD5 hash.
`AxRawUrlDecode(str)`	String	URL-decodes a string, converting `+` to space first.
`AxRgbToHex(r, g, b)`	String	Converts RGB values to uppercase `#RRGGBB` hex.
`AxSHA1(str)`	String	40-character lowercase SHA-1 hash.
`AxStripTags(str)`	String	Removes all HTML and XML tags from a string.
`AxUrlDecode(str)`	String	Percent-decodes a URL-encoded string (RFC 3986).

37.8 Validation and Type Inspection

Method	Returns	Description
`AxCtypeAlnum(str)`	Boolean	`True` if all characters are ASCII alphanumeric. Empty string returns `False`.
`AxCtypeAlpha(str)`	Boolean	`True` if all characters are ASCII letters. Empty string returns `False`.
`AxEmpty(value)`	Boolean	`True` if value is `Empty`,

Method	Returns	Description
		`Null`, `""`, `0`, `0.0`, or `False`.
`AxFilterValidateEmail(email)`	Boolean	`True` if the string is a syntactically valid email address.
`AxFilterValidateIp(ip)`	Boolean	`True` if the string is a valid IPv4 or IPv6 address.
`AxIsFloat(value)`	Boolean	`True` if the VM internal type is `VTDouble`.
`AxIsInt(value)`	Boolean	`True` if the VM internal type is `VTInteger`.
`AxIsSet(value)`	Boolean	`True` if value is not `Empty` or `Null`.

37.9 Date, Time, and Network

Method	Returns	Description
`AxDate(format [, timestamp])`	String	Formats a Unix timestamp using PHP-compatible tokens.
`AxGenerateGuid()`	String	RFC 4122 version 4 GUID string.
`AxGetRemoteFile(url)`	String/Boolean	HTTP GET response body on success, or `False` on failure.
`AxLastModified()`	Integer	Unix timestamp of the current script file's last modification.
`AxTime()`	Integer	Current Unix timestamp in seconds.

37.10 Remarks

- All method names are case-insensitive.
- Instantiate the library with `Server.CreateObject("G3AXON.FUNCTIONS")` before calling any method.

- Methods that perform file system or OS operations may behave differently on Windows versus Unix-like platforms. Refer to individual method pages for platform notes.

38 Properties

38.1 Overview

This page lists properties exposed by G3AXON.Functions.

38.2 Property List

38.3 Remarks

Read-only properties reject assignment and raise runtime errors when not writable.

39 AxEngineName

39.1 Overview

Use `AxEngineName` to identify the runtime engine name exposed by G3Pix AxonASP.

39.2 Prerequisites

- Instantiate the library with `Server.CreateObject("G3AXON.FUNCTIONS")`.

39.3 Syntax

```
engineName = obj.AxEngineName()
```

39.4 Parameters

- This method does not require parameters.

39.5 Return Value

- **String**: Always returns `AxonASP`.

39.6 Remarks

- Use this method for runtime identification checks before calling engine-specific routines.
- Method names are case-insensitive in VBScript dispatch.

39.7 Example

```
<%
Option Explicit
Dim ax, engineName

Set ax = Server.CreateObject("G3AXON.FUNCTIONS")
```

```
engineName = ax.AxEngineName()
If engineName = "AxonASP" Then
    Response.Write "Running on G3Pix AxonASP"
Else
    Response.Write "Unknown engine"
End If

Set ax = Nothing
%>
```

39.8 API Reference

- **Object**: `G3AXON.FUNCTIONS`
- **Method**: `AxEngineName`
- **Arguments**: `none`
- **Returns**: `String` (AxonASP)

40 AxVersion

40.1 Overview

Use `AxVersion` to read the active runtime version string of G3Pix AxonASP.

40.2 Prerequisites

- Instantiate the library with `Server.CreateObject("G3AXON.FUNCTIONS")`.

40.3 Syntax

```
versionText = obj.AxVersion()
```

40.4 Parameters

- This method does not require parameters.

40.5 Return Value

- **String**: Returns the current AxonASP runtime version (for example, `1.2.3` or a build-specific version string configured at build time).

40.6 Remarks

- Use this method for diagnostics, telemetry, and feature gating by runtime version.
- Method names are case-insensitive in VBScript dispatch.

40.7 Example

```
<%
Option Explicit
```

```
Dim ax, versionText

Set ax = Server.CreateObject("G3AXON.FUNCTIONS")

versionText = ax.AxVersion()
Response.Write "G3Pix AxonASP Version: " & versionText

Set ax = Nothing
%>
```

40.8 API Reference

- **Object**: `G3AXON.FUNCTIONS`
- **Method**: `AxVersion`
- **Arguments**: none
- **Returns**: `String` (current runtime version)

41 Get Runtime Diagnostic Report

41.1 Overview

Returns a multi-section diagnostic report covering runtime details, server context, a memory snapshot, all loaded configuration keys from `config/axonasp.toml`, and the AxonASP legal attribution block.

41.2 Prerequisites

Instantiate the library with `Server.CreateObject("G3AXON.FUNCTIONS")`.

41.3 Syntax

```
result = ax.AxRuntimeInfo()
```

41.4 Parameters

This method does not accept any parameters.

41.5 Return Value

- **String**: A plain-text diagnostic report. The format mirrors a `phpinfo()`-style output and is intended to be wrapped in a `<pre>` element for display.

41.6 Remarks

- Only expose this method on secured administrative pages. The report includes internal configuration values.
- Method names are case-insensitive.

41.7 Example

```
<%
Option Explicit
Dim ax, report
Set ax = Server.CreateObject("G3AXON.FUNCTIONS")

report = ax.AxRuntimeInfo()
Response.Write "<pre>" & Server.HTMLEncode(report) & "</pre>"

Set ax = Nothing
%>
```

41.8 API Reference

- **Object**: `G3AXON.FUNCTIONS`
- **Method**: `AxRuntimeInfo`
- **Arguments**: None
- **Returns**: String — multi-section plain-text diagnostic report

```
# AxGetEnv

## Overview

Use `AxGetEnv` to read one operating-system environment variable from the
 host process.

## Prerequisites

- Instantiate the library with `Server.CreateObject("G3AXON.FUNCTIONS")`.

## Syntax

```asp
value = obj.AxGetEnv(name)
```

41.9 Parameters

- **name** (String): Environment variable name to resolve.

41.10 Return Value

- **String**: Returns the value of the requested environment variable.
- **String**: Returns an empty string when the variable does not exist.
- **String**: Returns an empty string when `name` is omitted.
```

41.11 Remarks

- On Windows hosts, environment variable lookup is case-insensitive.
- On Unix-like hosts, environment variable lookup is case-sensitive.
- Method names are case-insensitive in VBScript dispatch.

41.12 Example

```
<%
Option Explicit
Dim ax, pathValue, missingValue

Set ax = Server.CreateObject("G3AXON.FUNCTIONS")

pathValue = ax.AxGetEnv("PATH")
missingValue = ax.AxGetEnv("AXONASP_VAR_THAT_DOES_NOT_EXIST")

Response.Write "PATH length: " & Len(pathValue) & "<br>"
Response.Write "Missing variable value: [" & missingValue & "]"

Set ax = Nothing
%>
```

41.13 API Reference

- **Object**: `G3AXON.FUNCTIONS`
- **Method**: `AxGetEnv`
- **Arguments**: `name As String`
- **Returns**: `String` (environment variable value, or empty string)

42 AxGetConfig

42.1 Overview

Use `AxGetConfig` to read one key from the active G3Pix AxonASP configuration.

42.2 Prerequisites

- Instantiate the library with `Server.CreateObject("G3AXON.FUNCTIONS")`.
- Ensure the runtime can load `config/axonasp.toml`.

42.3 Syntax

```
value = obj.AxGetConfig(key)
```

42.4 Parameters

- **key** (String): Fully qualified configuration key, such as `global.golang_memory_limit_mb`.

42.5 Return Value

- **Empty**: Returned when key is omitted.
- **Empty**: Returned when key is blank.
- **Empty**: Returned when the config file is not loaded.
- **Empty**: Returned when the key does not exist.
- **String**: Returned when the resolved config value is textual.
- **Boolean**: Returned when the resolved config value is true/false.
- **Integer**: Returned when the resolved config value is an integer type.
- **Double**: Returned when the resolved config value is a floating-point type.
- **Array**: Returned when the resolved config value is a list.

42.6 Remarks

- When `global.viper_automatic_env` is enabled, environment variables can override file values.
- Method names are case-insensitive in VBScript dispatch.

42.7 Example

```
<%
Option Explicit
Dim ax, memLimit, missingKey

Set ax = Server.CreateObject("G3AXON.FUNCTIONS")

memLimit = ax.AxGetConfig("global.golang_memory_limit_mb")
missingKey = ax.AxGetConfig("global.this_key_does_not_exist")

Response.Write "Memory limit: " & CStr(memLimit) & "<br>"
If IsEmpty(missingKey) Then
    Response.Write "Missing key returned Empty"
End If

Set ax = Nothing
%>
```

42.8 API Reference

- **Object**: `G3AXON.FUNCTIONS`
- **Method**: `AxGetConfig`
- **Arguments**: `key As String`
- **Returns**: `Empty`, `String`, `Boolean`, `Integer`, `Double`, or `Array` based on the resolved key state and value type

43 AxGetConfigKeys

43.1 Overview

Use `AxGetConfigKeys` to list every configuration key currently visible to G3Pix AxonASP.

43.2 Prerequisites

- Instantiate the library with `Server.CreateObject("G3AXON.FUNCTIONS")`.
- Ensure the runtime can load `config/axonasp.toml` when you expect non-empty output.

43.3 Syntax

```
keys = obj.AxGetConfigKeys()
```

43.4 Parameters

- This method does not require parameters.

43.5 Return Value

- **Array**: Zero-based VBArray containing configuration keys as strings.
- **Array**: Returns an empty zero-length array when no keys are available (for example, when the config file is not loaded).

43.6 Remarks

- Each element is a full key name (for example, `global.golang_memory_limit_mb`).
- Method names are case-insensitive in VBScript dispatch.

43.7 Example

```
<%
Option Explicit
Dim ax, keys, i

Set ax = Server.CreateObject("G3AXON.FUNCTIONS")

keys = ax.AxGetConfigKeys()
For i = 0 To UBound(keys)
    Response.Write keys(i) & "<br>"
Next

Set ax = Nothing
%>
```

43.8 API Reference

- **Object**: `G3AXON.FUNCTIONS`
- **Method**: `AxGetConfigKeys`
- **Arguments**: none
- **Returns**: `Array` of `String` keys (zero-based VBArray)

44 Get the User Home Directory Path

44.1 Overview

Returns the absolute path of the home directory for the user running the AxonASP process.

44.2 Prerequisites

Instantiate the library with `Server.CreateObject("G3AXON.FUNCTIONS")`.

44.3 Syntax

```
result = obj.AxUserHomeDirPath()
```

44.4 Parameters

This method does not require parameters.

44.5 Return Value

- **String**: Returns the absolute path of the current user's home directory.
- **String**: Returns an empty string when the home directory cannot be determined.

44.6 Remarks

- On Windows, falls back to `USERPROFILE` then `HOME` environment variables.
- On Unix-like systems, falls back to the `HOME` environment variable.
- Method names are case-insensitive in VBScript dispatch.

44.7 Example

```
<%
Option Explicit
Dim ax
Set ax = Server.CreateObject("G3AXON.FUNCTIONS")

Response.Write "Home: " & ax.AxUserHomeDirPath()

Set ax = Nothing
%>
```

44.8 API Reference

- **Object**: `G3AXON.FUNCTIONS`
- **Method**: `AxUserHomeDirPath`
- **Arguments**: `none`
- **Returns**: `String` (absolute home directory path)

45 Get the Configuration File Path

45.1 Overview

Returns the resolved absolute path to the AxonASP configuration file (`config/axonasp.toml`).

45.2 Prerequisites

Instantiate the library with `Server.CreateObject("G3AXON.FUNCTIONS")`.

45.3 Syntax

```
result = obj.AxUserConfigDirPath()
```

45.4 Parameters

This method does not require parameters.

45.5 Return Value

- **String**: Returns the absolute path to `config/axonasp.toml`, resolved from common search locations relative to the executable.

45.6 Remarks

- The path is resolved by searching `./config/axonasp.toml`, `../config/axonasp.toml`, and the directory of the running executable.
- Method names are case-insensitive in VBScript dispatch.

45.7 Example

```
<%
Option Explicit
Dim ax
Set ax = Server.CreateObject("G3AXON.FUNCTIONS")

Response.Write "Config path: " & ax.AxUserConfigDirPath()

Set ax = Nothing
%>
```

45.8 API Reference

- **Object**: `G3AXON.FUNCTIONS`
- **Method**: `AxUserConfigDirPath`
- **Arguments**: `none`
- **Returns**: `String` (absolute path to `config/axonasp.toml`)

46 Get the Cache Directory Path

46.1 Overview

Returns the absolute path to the AxonASP cache directory (`.temp/cache/`), including a trailing path separator.

46.2 Prerequisites

Instantiate the library with `Server.CreateObject("G3AXON.FUNCTIONS")`.

46.3 Syntax

```
result = obj.AxCacheDirPath()
```

46.4 Parameters

This method does not require parameters.

46.5 Return Value

- **String**: Returns the absolute path to the cache directory, always ending with a path separator character.

46.6 Remarks

- Method names are case-insensitive in VBScript dispatch.

46.7 Example

```
<%
Option Explicit
Dim ax, cachePath
Set ax = Server.CreateObject("G3AXON.FUNCTIONS")

cachePath = ax.AxCacheDirPath()
Response.Write "Cache: " & cachePath
' Example output on Windows: C:\axonasp\.temp\cache\

Set ax = Nothing
%>
```

46.8 API Reference

- **Object**: `G3AXON.FUNCTIONS`
- **Method**: `AxCacheDirPath`
- **Arguments**: `none`
- **Returns**: `String` (absolute path to `.temp/cache/` with trailing separator)

47 Check If a Character Is a Path Separator

47.1 Overview

Returns `True` when the provided single character is a valid path separator for the current platform.

47.2 Prerequisites

Instantiate the library with `Server.CreateObject("G3AXON.FUNCTIONS")`.

47.3 Syntax

```
result = obj.AxIsPathSeparator(character)
```

47.4 Parameters

Parameter	Type	Required	Description
character	String	Yes	A single character to test. Multi-character strings or empty strings always return `False`.

47.5 Return Value

- **Boolean**: Returns `True` when `character` is a valid path separator on the current operating system.
- **Boolean**: Returns `False` when `character` is not a path separator, is empty, is multi-character, or no argument is provided.

47.6 Remarks

- On Windows, both `\` and `/` may be valid path separators.
- Method names are case-insensitive in VBScript dispatch.

47.7 Example

```
<%
Option Explicit
Dim ax
```

```
Set ax = Server.CreateObject("G3AXON.FUNCTIONS")

Response.Write CStr(ax.AxIsPathSeparator("/")) & "<br>" ' True on Unix, p
ossibly True on Windows
Response.Write CStr(ax.AxIsPathSeparator("a")) & "<br>" ' False

Set ax = Nothing
%>
```

47.8 API Reference

- **Object**: `G3AXON.FUNCTIONS`
- **Method**: `AxIsPathSeparator`
- **Arguments**: `character As String`
- **Returns**: `Boolean` (`True` if the character is a valid path separator)

48 Change File Timestamps

48.1 Overview

Changes the access and modification timestamps of a file using Unix epoch seconds.

48.2 Prerequisites

Instantiate the library with `Server.CreateObject("G3AXON.FUNCTIONS")`.

48.3 Syntax

```
result = obj.AxChangeTimes(path, accessTime, modifyTime)
```

48.4 Parameters

Parameter	Type	Required	Description
path	String	Yes	The file path whose timestamps will be updated.
accessTime	Integer	Yes	The new access time as a Unix epoch timestamp (seconds since 1970-01-01 UTC).
modifyTime	Integer	Yes	The new modification time as a Unix epoch timestamp.

48.5 Return Value

- **Boolean**: Returns `True` when the timestamps were changed successfully.
- **Boolean**: Returns `False` when the path is empty, the file does not exist, or the operation fails due to permissions.

48.6 Remarks

- Requires at least three arguments; returns `False` with fewer.
- Method names are case-insensitive in VBScript dispatch.

48.7 Example

```
<%
Option Explicit
Dim ax, ok
Set ax = Server.CreateObject("G3AXON.FUNCTIONS")

ok = ax.AxChangeTimes("C:\\temp\\file.txt", 1700000000, 1700000001)
If ok Then
    Response.Write "Timestamps updated."
End If

Set ax = Nothing
%>
```

48.8 API Reference

- **Object**: `G3AXON.FUNCTIONS`
- **Method**: `AxChangeTimes`
- **Arguments**: `path As String, accessTime As Integer, modifyTime As Integer`
- **Returns**: `Boolean` (`True` on success, `False` on failure)

49 Change File Mode Permissions

49.1 Overview

Changes the mode (permission bits) of a file using an octal string (for example, `"0644"`).

49.2 Prerequisites

Instantiate the library with `Server.CreateObject("G3AXON.FUNCTIONS")`.

49.3 Syntax

```
result = obj.AxChangeMode(path, mode)
```

49.4 Parameters

Parameter	Type	Required	Description
path	String	Yes	The file path to modify.
mode	String	Yes	The octal permission string (for example, `"0644"`, `"0755"`).

49.5 Return Value

- **Boolean**: Returns `True` when the mode was applied successfully.
- **Boolean**: Returns `False` when fewer than two arguments are provided, the path or mode is empty, the octal string is invalid, or the operation fails due to permissions.

49.6 Remarks

- This method applies the Unix-style file mode bits. On Windows, the effect is limited; only the read-only attribute is affected.
- Method names are case-insensitive in VBScript dispatch.

49.7 Example

```
<%
Option Explicit
Dim ax
Set ax = Server.CreateObject("G3AXON.FUNCTIONS")

If ax.AxChangeMode("/var/www/upload.log", "0644") Then
    Response.Write "Permissions updated."
End If

Set ax = Nothing
%>
```

49.8 API Reference

- **Object**: `G3AXON.FUNCTIONS`
- **Method**: `AxChangeMode`
- **Arguments**: `path As String, mode As String`
- **Returns**: `Boolean` (`True` on success, `False` on failure)

50 Create a Hard Link

50.1 Overview

Creates a hard link at `linkPath` that points to the existing file at `sourcePath`.

50.2 Prerequisites

Instantiate the library with `Server.CreateObject("G3AXON.FUNCTIONS")`.

50.3 Syntax

```
result = obj.AxCreateLink(sourcePath, linkPath)
```

50.4 Parameters

Parameter	Type	Required	Description
sourcePath	String	Yes	The path of the existing file to link to.
linkPath	String	Yes	The path of the new hard link to create.

50.5 Return Value

- **Boolean**: Returns `True` when the hard link was created successfully.
- **Boolean**: Returns `False` when fewer than two arguments are provided, either path is empty, or the operation fails.

50.6 Remarks

- Hard links share the same inode as the source; deleting one does not remove the other.
- Some restricted environments or file system types may deny link creation.
- Method names are case-insensitive in VBScript dispatch.

50.7 Example

```
<%
Option Explicit
Dim ax
Set ax = Server.CreateObject("G3AXON.FUNCTIONS")

If ax.AxCreateLink("/var/data/original.txt", "/var/data/backup.txt") Then
    Response.Write "Hard link created."
End If
```

```
Set ax = Nothing
%>
```

50.8 API Reference

- **Object**: `G3AXON.FUNCTIONS`
- **Method**: `AxCreateLink`
- **Arguments**: `sourcePath As String, linkPath As String`
- **Returns**: `Boolean` (`True` on success, `False` on failure)

51 Change File Owner and Group

51.1 Overview

Changes the owner and group of a file by specifying numeric user ID (uid) and group ID (gid).

51.2 Prerequisites

Instantiate the library with `Server.CreateObject("G3AXON.FUNCTIONS")`.

51.3 Syntax

```
result = obj.AxChangeOwner(path, uid, gid)
```

51.4 Parameters

Parameter	Type	Required	Description
path	String	Yes	The file path to change ownership of.
uid	Integer	Yes	The numeric user ID of the new owner.
gid	Integer	Yes	The numeric group ID of the new owner group.

51.5 Return Value

- **Boolean**: Returns `True` when ownership was changed successfully.
- **Boolean**: Returns `False` when fewer than three arguments are provided, the path is empty, or the operation fails due to permissions.

51.6 Remarks

- On Windows and non-privileged Unix environments, this method commonly returns `False`.
- Typically requires root/administrator privileges on Unix-like systems.
- Method names are case-insensitive in VBScript dispatch.

51.7 Example

```
<%
Option Explicit
Dim ax
Set ax = Server.CreateObject("G3AXON.FUNCTIONS")

' Change owner to root:root (uid=0, gid=0)
If ax.AxChangeOwner("/var/data/file.txt", 0, 0) Then
    Response.Write "Owner changed."
Else
    Response.Write "Failed (insufficient privileges)."
End If

Set ax = Nothing
%>
```

51.8 API Reference

- **Object**: `G3AXON.FUNCTIONS`
- **Method**: `AxChangeOwner`
- **Arguments**: `path As String, uid As Integer, gid As Integer`
- **Returns**: `Boolean` (`True` on success, `False` on failure)

52 Shut Down the AxonASP Server

52.1 Overview

Immediately terminates the AxonASP server process. This method requires the shutdown function to be enabled in the configuration file before it can execute.

52.2 Prerequisites

Instantiate the library with `Server.CreateObject("G3AXON.FUNCTIONS")`.

Enable the shutdown function in `config/axonasp.toml`:

```
[axfunctions]
enable_axservershutdown_function = true
```

52.3 Syntax

```
result = ax.AxShutdownAxonASPServer()
```

52.4 Parameters

This method does not accept any parameters.

52.5 Return Value

- **Boolean `True`:** The shutdown was triggered and the server process will terminate immediately.
- **Boolean `False`:** The `enable_axservershutdown_function` configuration key is `false` or absent. The server continues running.

52.6 Remarks

- **This call is irreversible.** All in-flight requests are immediately cancelled when the server exits.
- Only expose this method in a secured administrative page. Do not make it publicly accessible.
- The method calls `os.Exit` internally. No further code in the current script executes after a `True` return.
- Method names are case-insensitive.

52.7 Example

```
<%
Option Explicit
Dim ax, result
Set ax = Server.CreateObject("G3AXON.FUNCTIONS")

result = ax.AxShutdownAxonASPServer()

If result Then
    ' Execution never reaches here — server is terminating.
Else
    Response.Write "Shutdown is not enabled in the configuration."
End If

Set ax = Nothing
%>
```

52.8 API Reference

- **Object:** `G3AXON.FUNCTIONS`
- **Method:** `AxShutdownAxonASPServer`
- **Arguments:** None
- **Returns:** Boolean — `True` if the server terminated; `False` if shutdown is disabled in configuration

53 Change the Current Working Directory

53.1 Overview

Changes the current working directory of the AxonASP process to the specified path.

53.2 Prerequisites

Instantiate the library with `Server.CreateObject("G3AXON.FUNCTIONS")`.

53.3 Syntax

```
result = obj.AxChangeDir(path)
```

53.4 Parameters

Parameter	Type	Required	Description
path	String	Yes	The absolute or relative path of the directory to switch to.

53.5 Return Value

- **Boolean**: Returns `True` when the directory was changed successfully.
- **Boolean**: Returns `False` when the path is invalid, inaccessible, or no argument is provided.

53.6 Remarks

- Changing the working directory affects the entire process. In a multi-threaded web server environment, use this function with caution because it may interfere with other concurrent requests.
- Method names are case-insensitive in VBScript dispatch.

53.7 Example

```
<%
Option Explicit
Dim ax, ok
Set ax = Server.CreateObject("G3AXON.FUNCTIONS")

ok = ax.AxChangeDir("C:\Windows\Temp")
If ok Then
    Response.Write "Directory changed to: " & ax.AxCurrentDir()
Else
    Response.Write "Failed to change directory."
End If
```

```
Set ax = Nothing
%>
```

53.8 API Reference

- **Object**: `G3AXON.FUNCTIONS`
- **Method**: `AxChangeDir`
- **Arguments**: `path As String`
- **Returns**: `Boolean` (`True` on success, `False` on failure) %›

```
# Get the Current Working Directory

## Overview

Returns the absolute path of the current working directory of the AxonASP
 process.

## Prerequisites

Instantiate the library with `Server.CreateObject("G3AXON.FUNCTIONS")`.

## Syntax

```asp
result = obj.AxCurrentDir()
```

53.9 Parameters

This method does not require parameters.

53.10 Return Value

- **String**: Returns the absolute path of the current working directory.
- **String**: Returns an empty string when an OS error occurs.

53.11 Remarks

- Method names are case-insensitive in VBScript dispatch.

53.12 Example

<%
Option Explicit
Dim ax
Set ax = Server.CreateObject("G3AXON.FUNCTIONS")

Response.Write "Working directory: " & ax.AxCurrentDir()
```

```
Set ax = Nothing
%>
```

53.13 API Reference

- **Object**: `G3AXON.FUNCTIONS`
- **Method**: `AxCurrentDir`
- **Arguments**: none
- **Returns**: `String` (absolute path of the current working directory)

54 Get the Machine Hostname

54.1 Overview

Returns the network hostname of the machine where the AxonASP process is running.

54.2 Prerequisites

Instantiate the library with `Server.CreateObject("G3AXON.FUNCTIONS")`.

54.3 Syntax

```
result = obj.AxHostnameValue()
```

54.4 Parameters

This method does not require parameters.

54.5 Return Value

- **String**: Returns the system hostname as reported by the operating system.
- **String**: Returns an empty string when the hostname cannot be determined.

54.6 Remarks

- Method names are case-insensitive in VBScript dispatch.

54.7 Example

```
<%
Option Explicit
Dim ax
Set ax = Server.CreateObject("G3AXON.FUNCTIONS")

Response.Write "Hostname: " & ax.AxHostnameValue()

Set ax = Nothing
%>
```

54.8 API Reference

- **Object**: `G3AXON.FUNCTIONS`
- **Method**: `AxHostnameValue`
- **Arguments**: `none`
- **Returns**: `String` (machine hostname, or empty string)

55 Clear All Environment Variables

55.1 Overview

Removes all environment variables from the current AxonASP process environment.

55.2 Prerequisites

Instantiate the library with `Server.CreateObject("G3AXON.FUNCTIONS")`.

55.3 Syntax

```
result = obj.AxClearEnvironment()
```

55.4 Parameters

This method does not require parameters.

55.5 Return Value

- **Boolean**: Always returns `True` after execution.

55.6 Remarks

- This is a destructive, process-wide operation. All inherited and set environment variables are removed.
- Use with extreme care because system utilities and libraries may fail if they depend on specific environment variables.
- Method names are case-insensitive in VBScript dispatch.

55.7 Example

```
<%
Option Explicit
Dim ax
Set ax = Server.CreateObject("G3AXON.FUNCTIONS")

' WARNING: Clears ALL environment variables from the process
ax.AxClearEnvironment()
Response.Write "Environment cleared."

Set ax = Nothing
%>
```

55.8 API Reference

- **Object**: `G3AXON.FUNCTIONS`
- **Method**: `AxClearEnvironment`
- **Arguments**: `none`
- **Returns**: `Boolean` (always `True`)

```
# List All Environment Variables

## Overview

Returns a snapshot of all environment variables currently defined in the
process as a VBArray.

## Prerequisites

Instantiate the library with `Server.CreateObject("G3AXON.FUNCTIONS")`.

## Syntax

```asp
result = obj.AxEnvironmentList()
```

55.9 Parameters

This method does not require parameters.

55.10 Return Value

- **Array**: Returns a zero-based VBArray of String values, each in `KEY=VALUE` format.

55.11 Remarks

- Pseudo-environment entries (such as entries starting with `=` on Windows) are filtered out automatically.
- The order of elements depends on the operating system.
- Method names are case-insensitive in VBScript dispatch.

55.12 Example

```asp
<%
Option Explicit
Dim ax, envList, i
Set ax = Server.CreateObject("G3AXON.FUNCTIONS")

envList = ax.AxEnvironmentList()
For i = 0 To UBound(envList)
```
```

```
    Response.Write Server.HTMLEncode(envList(i)) & "<br>"
Next

Set ax = Nothing
%>
```

55.13 API Reference

- **Object**: `G3AXON.FUNCTIONS`
- **Method**: `AxEnvironmentList`
- **Arguments**: none
- **Returns**: `Array` (zero-based VBArray of `KEY=VALUE` strings)

56 Get an Environment Variable Value

56.1 Overview

Retrieves the value of a specific environment variable from the process, with an optional default fallback.

56.2 Prerequisites

Instantiate the library with `Server.CreateObject("G3AXON.FUNCTIONS")`.

56.3 Syntax

```
result = obj.AxEnvironmentValue(name [, default])
```

56.4 Parameters

Parameter	Type	Required	Description
name	String	Yes	The name of the environment variable to look up.
default	Variant	Optional	The value to return when the variable is not found. Defaults to an empty string.

56.5 Return Value

- **String**: Returns the value of the environment variable when it exists.
- **Variant**: Returns `default` when the variable is not found and a default was provided.
- **String**: Returns an empty string when the variable is not found and no default was provided.

56.6 Remarks

- On Windows, lookup is case-insensitive. On Unix-like systems, lookup is case-sensitive.
- Unlike `AxGetEnv`, this method lets you specify a fallback value in the same call.
- Method names are case-insensitive in VBScript dispatch.

56.7 Example

```
<%
Option Explicit
Dim ax, port
Set ax = Server.CreateObject("G3AXON.FUNCTIONS")

port = ax.AxEnvironmentValue("APP_PORT", "8080")
Response.Write "Port: " & port

Set ax = Nothing
%>
```

56.8 API Reference

- **Object**: `G3AXON.FUNCTIONS`
- **Method**: `AxEnvironmentValue`
- **Arguments**: `name As String [, default As Variant]`
- **Returns**: `String` (variable value or default)

57 Get the Current Process ID

57.1 Overview

Returns the operating system Process ID (PID) of the current AxonASP process.

57.2 Prerequisites

Instantiate the library with `Server.CreateObject("G3AXON.FUNCTIONS")`.

57.3 Syntax

```
result = obj.AxProcessId()
```

57.4 Parameters

This method does not require parameters.

57.5 Return Value

- **Integer**: Returns the numeric PID assigned by the OS to the current process.

57.6 Remarks

- Method names are case-insensitive in VBScript dispatch.

57.7 Example

```
<%
Option Explicit
Dim ax
Set ax = Server.CreateObject("G3AXON.FUNCTIONS")

Response.Write "PID: " & ax.AxProcessId()

Set ax = Nothing
%>
```

57.8 API Reference

- **Object**: `G3AXON.FUNCTIONS`
- **Method**: `AxProcessId`
- **Arguments**: none
- **Returns**: `Integer` (operating system process identifier)

58 Get the Effective User ID

58.1 Overview

Returns the effective user ID (`euid`) of the AxonASP process.

58.2 Prerequisites

Instantiate the library with `Server.CreateObject("G3AXON.FUNCTIONS")`.

58.3 Syntax

```
result = obj.AxEffectiveUserId()
```

58.4 Parameters

This method does not require parameters.

58.5 Return Value

- **Integer**: Returns the numeric effective user ID on Unix-like systems.
- **Integer**: Returns `-1` on Windows, where the effective user ID concept does not apply.

58.6 Remarks

- On Unix-like systems, a return value of `0` indicates the process is running as root.
- Method names are case-insensitive in VBScript dispatch.

58.7 Example

```
<%
Option Explicit
Dim ax
Set ax = Server.CreateObject("G3AXON.FUNCTIONS")

If ax.AxEffectiveUserId() = 0 Then
    Response.Write "Running as root/superuser"
ElseIf ax.AxEffectiveUserId() = -1 Then
    Response.Write "Windows: effective UID not applicable"
Else
    Response.Write "Effective UID: " & ax.AxEffectiveUserId()
End If

Set ax = Nothing
%>
```

58.8 API Reference

- **Object**: `G3AXON.FUNCTIONS`
- **Method**: `AxEffectiveUserId`
- **Arguments**: none
- **Returns**: `Integer` (euid on Unix, `-1` on Windows)

59 Get the Directory Separator Character

59.1 Overview

Returns the character the operating system uses to separate directory components in a file path.

59.2 Prerequisites

Instantiate the library with `Server.CreateObject("G3AXON.FUNCTIONS")`.

59.3 Syntax

```
result = obj.AxDirSeparator()
```

59.4 Parameters

This method does not require parameters.

59.5 Return Value

- **String**: Returns `"\"` on Windows systems.
- **String**: Returns `"/"` on Unix-like systems.

59.6 Remarks

- Method names are case-insensitive in VBScript dispatch.

59.7 Example

```
<%
Option Explicit
Dim ax, sep
Set ax = Server.CreateObject("G3AXON.FUNCTIONS")

sep = ax.AxDirSeparator()
Response.Write "docs" & sep & "manual" & sep & "index.html"

Set ax = Nothing
%>
```

59.8 API Reference

- **Object**: `G3AXON.FUNCTIONS`
- **Method**: `AxDirSeparator`
- **Arguments**: none
- **Returns**: `String` (\ on Windows, / on Unix)

60 Get the Path List Separator Character

60.1 Overview

Returns the character the operating system uses to separate entries in a path list, such as the `PATH` environment variable.

60.2 Prerequisites

Instantiate the library with `Server.CreateObject("G3AXON.FUNCTIONS")`.

60.3 Syntax

```
result = ax.AxPathListSeparator()
```

60.4 Parameters

This method does not accept any parameters.

60.5 Return Value

- **String**: `";"` on Windows. `":"` on Unix-like systems.

60.6 Remarks

- Use this method to build portable path list strings without hardcoding platform-specific separator characters.
- Method names are case-insensitive.

60.7 Example

```
<%
Option Explicit
Dim ax, sep, pathList
Set ax = Server.CreateObject("G3AXON.FUNCTIONS")

sep = ax.AxPathListSeparator()
pathList = "/usr/bin" & sep & "/usr/local/bin" & sep & "/opt/go/bin"

Response.Write "PATH list: " & Server.HTMLEncode(pathList)

Set ax = Nothing
%>
```

60.8 API Reference

- **Object**: `G3AXON.FUNCTIONS`
- **Method**: `AxPathListSeparator`
- **Arguments**: None
- **Returns**: String — `";"` on Windows, `":"` on Unix

61 Get the Native Integer Size in Bytes

61.1 Overview

Returns the number of bytes used to store a native integer on the current platform.

61.2 Prerequisites

Instantiate the library with `Server.CreateObject("G3AXON.FUNCTIONS")`.

61.3 Syntax

```
result = obj.AxIntegerSizeBytes()
```

61.4 Parameters

This method does not require parameters.

61.5 Return Value

- **Integer**: Returns 4 on 32-bit systems and 8 on 64-bit systems.

61.6 Remarks

- Method names are case-insensitive in VBScript dispatch.

61.7 Example

```
<%
Option Explicit
Dim ax
Set ax = Server.CreateObject("G3AXON.FUNCTIONS")

Response.Write ax.AxIntegerSizeBytes() & " bytes"
' Output: 8 bytes (on a 64-bit system)

Set ax = Nothing
%>
```

61.8 API Reference

- **Object**: `G3AXON.FUNCTIONS`
- **Method**: `AxIntegerSizeBytes`
- **Arguments**: none
- **Returns**: `Integer` (4 on 32-bit, 8 on 64-bit)

62 Get the Platform Architecture Bit Width

62.1 Overview

Returns the native integer bit size of the current platform (typically 32 or 64).

62.2 Prerequisites

Instantiate the library with `Server.CreateObject("G3AXON.FUNCTIONS")`.

62.3 Syntax

```
result = obj.AxPlatformBits()
```

62.4 Parameters

This method does not require parameters.

62.5 Return Value

- **Integer**: Returns 32 on 32-bit systems and 64 on 64-bit systems.

62.6 Remarks

- Method names are case-insensitive in VBScript dispatch.

62.7 Example

```
<%
Option Explicit
Dim ax
Set ax = Server.CreateObject("G3AXON.FUNCTIONS")

Response.Write ax.AxPlatformBits() & "-bit"
' Output: 64-bit

Set ax = Nothing
%>
```

62.8 API Reference

- **Object**: `G3AXON.FUNCTIONS`
- **Method**: `AxPlatformBits`
- **Arguments**: none
- **Returns**: `Integer` (32 or 64)

63 Get the Executable Path

63.1 Overview

Returns the absolute path of the AxonASP executable file currently running the process.

63.2 Prerequisites

Instantiate the library with `Server.CreateObject("G3AXON.FUNCTIONS")`.

63.3 Syntax

```
result = obj.AxExecutablePath()
```

63.4 Parameters

This method does not require parameters.

63.5 Return Value

- **String**: Returns the absolute path to the running executable file.
- **String**: Returns an empty string when the path cannot be determined.

63.6 Remarks

- Method names are case-insensitive in VBScript dispatch.

63.7 Example

```
<%
Option Explicit
Dim ax
Set ax = Server.CreateObject("G3AXON.FUNCTIONS")

Response.Write "Executable: " & ax.AxExecutablePath()

Set ax = Nothing
%>
```

63.8 API Reference

- **Object**: `G3AXON.FUNCTIONS`
- **Method**: `AxExecutablePath`
- **Arguments**: none
- **Returns**: `String` (absolute path to the running executable)

64 Execute a Shell Command

64.1 Overview

Executes an external shell command and returns the combined stdout and stderr output as a string.

64.2 Prerequisites

Instantiate the library with `Server.CreateObject("G3AXON.FUNCTIONS")`.

64.3 Syntax

```
result = obj.AxExecute(command)
```

64.4 Parameters

Parameter	Type	Required	Description
command	String	Yes	The full command line to execute.

64.5 Return Value

- **String**: Returns the combined stdout and stderr output of the command, with trailing newlines stripped.
- **Boolean**: Returns `False` when `command` is empty or no argument is provided.

64.6 Remarks

- On Windows, the command runs via `cmd.exe /c`. On Unix-like systems, it runs via `sh -c`.
- This is a blocking call. The script waits until the command completes.
- Never pass user-provided input directly to this function without validation.
- Method names are case-insensitive in VBScript dispatch.

64.7 Example

```
<%
Option Explicit
Dim ax, output
Set ax = Server.CreateObject("G3AXON.FUNCTIONS")

output = ax.AxExecute("dir C:\\") ' Windows
Response.Write "<pre>" & Server.HTMLEncode(output) & "</pre>"

Set ax = Nothing
%>
```

64.8 API Reference

- **Object**: `G3AXON.FUNCTIONS`
- **Method**: `AxExecute`
- **Arguments**: `command As String`
- **Returns**: `String` (command output), or `Boolean False` when command is empty

65 Get System Information

65.1 Overview

Retrieves operating system and runtime environment details for the current server process.

65.2 Prerequisites

Instantiate the library with `Server.CreateObject("G3AXON.FUNCTIONS")`.

65.3 Syntax

```
result = ax.AxSystemInfo([mode])
```

65.4 Parameters

Parameter	Type	Required	Description
mode	String	No	A single character

Parameter	Type	Required	Description
			selecting the category of information to return. Omit or pass any unrecognized value to receive all categories combined.

mode values:

Value	Returns
`"s"`	Operating system name (e.g., `"windows"`, `"linux"`, `"darwin"`).
`"n"`	Hostname of the machine.
`"v"`	Go runtime version (e.g., `"go1.22.0"`).
`"m"`	Machine architecture (e.g., `"amd64"`, `"arm64"`).
`"a"` or omitted	All categories combined in a single string.

65.5 Return Value

- **String**: The requested system information. Returns all categories combined when `mode` is omitted or unrecognized.

65.6 Remarks

- Any unrecognized or empty mode value defaults to `"a"` (all information).
- Method names are case-insensitive.

65.7 Example

```
<%
Option Explicit
Dim ax
Set ax = Server.CreateObject("G3AXON.FUNCTIONS")

Response.Write "OS: " & ax.AxSystemInfo("s") & "<br>"
Response.Write "Hostname: " & ax.AxSystemInfo("n") & "<br>"
Response.Write "Architecture: " & ax.AxSystemInfo("m") & "<br>"
Response.Write "Go version: " & ax.AxSystemInfo("v") & "<br>"
Response.Write "Full info: " & ax.AxSystemInfo() & "<br>"
```

```
Set ax = Nothing
%>
```

65.8 API Reference

- **Object**: `G3AXON.FUNCTIONS`
- **Method**: `AxSystemInfo`
- **Arguments**: `mode` (String, optional) — `"s"`, `"n"`, `"v"`, `"m"`, or `"a"`
- **Returns**: String — system information for the selected category

66 Get the Current Process User Name

66.1 Overview

Returns the username of the operating system user running the current AxonASP process.

66.2 Prerequisites

Instantiate the library with `Server.CreateObject("G3AXON.FUNCTIONS")`.

66.3 Syntax

```
result = obj.AxCurrentUser()
```

66.4 Parameters

This method does not require parameters.

66.5 Return Value

- **String**: Returns the username of the process owner.
- **String**: Returns an empty string when the username cannot be determined.

66.6 Remarks

- On Windows, falls back to the `USERNAME` environment variable if the system API call fails.
- On Unix-like systems, falls back to the `USER` environment variable if the system API call fails.
- Method names are case-insensitive in VBScript dispatch.

66.7 Example

```
<%
Option Explicit
Dim ax
Set ax = Server.CreateObject("G3AXON.FUNCTIONS")
```

```
Response.Write "Process user: " & ax.AxCurrentUser()

Set ax = Nothing
%>
```

66.8 API Reference

- **Object**: `G3AXON.FUNCTIONS`
- **Method**: `AxCurrentUser`
- **Arguments**: none
- **Returns**: `String` (username of the process owner, or empty string)

67 Write HTML-Escaped Text to the Response

67.1 Overview

Writes a string to the current HTTP response after escaping all HTML special characters. This is a safe alternative to `Response.Write` when outputting user-supplied or untrusted data.

67.2 Prerequisites

Instantiate the library with `Server.CreateObject("G3AXON.FUNCTIONS")`.

67.3 Syntax

```
ax.AxW(text)
```

This method is also accessible as `ax.document.write(text)` and `ax.documentwrite(text)` for compatibility.

67.4 Parameters

Parameter	Type	Required	Description
text	String	Yes	The text to write to the response. HTML special characters are escaped before output.

67.5 Return Value

- **Empty**: This method does not return a value.

67.6 Remarks

- Characters escaped: `&` ⟶ `&`, `<` ⟶ `<`, `>` ⟶ `>`, `"` ⟶ `"`.

- Use `Response.Write` when you intentionally need to output raw HTML markup.
- Method names are case-insensitive.

67.7 Example

```
<%
Option Explicit
Dim ax, userInput
Set ax = Server.CreateObject("G3AXON.FUNCTIONS")

userInput = "<script>alert('xss')</script>"

' Renders safely as literal text — no script executes.
ax.AxW userInput

Set ax = Nothing
%>
```

67.8 API Reference

- **Object**: `G3AXON.FUNCTIONS`
- **Method**: `AxW` (also: `document.write`, `documentwrite`)
- **Arguments**: `text` (String, required)
- **Returns**: Empty

68 Return the Maximum Value

68.1 Overview

Returns the largest numeric value from all provided arguments.

68.2 Prerequisites

Instantiate the library with `Server.CreateObject("G3AXON.FUNCTIONS")`.

68.3 Syntax

```
result = obj.AxMax(n1, n2, ...)
```

68.4 Parameters

Parameter	Type	Required	Description
`n1`, `n2`, …	Double	Yes (at least one)	One or more numeric values to compare. All values are coerced to Double before comparison.

68.5 Return Value

- **Double**: Returns the largest value among all arguments.
- **Integer**: Returns `0` when no arguments are provided.

68.6 Remarks

- All values are coerced to Double before comparison.
- Method names are case-insensitive in VBScript dispatch.

68.7 Example

```
<%
Option Explicit
Dim ax
Set ax = Server.CreateObject("G3AXON.FUNCTIONS")

Response.Write ax.AxMax(10, 45.7, 32, -5)
' Output: 45.7

Response.Write ax.AxMax(100, 200, 150)
' Output: 200

Set ax = Nothing
%>
```

68.8 API Reference

- **Object**: `G3AXON.FUNCTIONS`
- **Method**: `AxMax`
- **Arguments**: `n1 As Double, n2 As Double, ...` (variadic)
- **Returns**: `Double` (largest value among arguments)

69 Return the Minimum Value

69.1 Overview

Returns the smallest numeric value from all provided arguments.

69.2 Prerequisites

Instantiate the library with `Server.CreateObject("G3AXON.FUNCTIONS")`.

69.3 Syntax

```
result = obj.AxMin(n1, n2, ...)
```

69.4 Parameters

Parameter	Type	Required	Description

Parameter	Type	Required	Description
n1, n2, …	Double	Yes (at least one)	One or more numeric values to compare. All values are coerced to Double before comparison.

69.5 Return Value

- **Double**: Returns the smallest value among all arguments.
- **Integer**: Returns `0` when no arguments are provided.

69.6 Remarks

- All values are coerced to Double before comparison.
- Method names are case-insensitive in VBScript dispatch.

69.7 Example

```
<%
Option Explicit
Dim ax
Set ax = Server.CreateObject("G3AXON.FUNCTIONS")

Response.Write ax.AxMin(10, 45, 32, -5.2)
' Output: -5.2

Response.Write ax.AxMin(100, 200, 150)
' Output: 100

Set ax = Nothing
%>
```

69.8 API Reference

- **Object**: `G3AXON.FUNCTIONS`
- **Method**: `AxMin`
- **Arguments**: `n1 As Double, n2 As Double, ...` (variadic)
- **Returns**: `Double` (smallest value among arguments)

70 Get the Maximum Platform Integer Value

70.1 Overview

Returns the maximum value that a signed 64-bit integer can hold on the current platform.

70.2 Prerequisites

Instantiate the library with `Server.CreateObject("G3AXON.FUNCTIONS")`.

70.3 Syntax

```
result = obj.AxIntegerMax()
```

70.4 Parameters

This method does not require parameters.

70.5 Return Value

- **Integer**: Returns `9,223,372,036,854,775,807` on 64-bit systems.

70.6 Remarks

- Method names are case-insensitive in VBScript dispatch.

70.7 Example

```
<%
Option Explicit
Dim ax
Set ax = Server.CreateObject("G3AXON.FUNCTIONS")

Response.Write ax.AxIntegerMax()
' Output: 9223372036854775807

Set ax = Nothing
%>
```

70.8 API Reference

- **Object**: `G3AXON.FUNCTIONS`
- **Method**: `AxIntegerMax`
- **Arguments**: `none`
- **Returns**: `Integer` (maximum signed 64-bit integer value)

71 Get the Minimum Platform Integer Value

71.1 Overview

Returns the minimum value that a signed 64-bit integer can hold on the current platform.

71.2 Prerequisites

Instantiate the library with `Server.CreateObject("G3AXON.FUNCTIONS")`.

71.3 Syntax

```
result = obj.AxIntegerMin()
```

71.4 Parameters

This method does not require parameters.

71.5 Return Value

- **Integer**: Returns `-9,223,372,036,854,775,808` on 64-bit systems.

71.6 Remarks

- Method names are case-insensitive in VBScript dispatch.

71.7 Example

```
<%
Option Explicit
Dim ax
Set ax = Server.CreateObject("G3AXON.FUNCTIONS")

Response.Write ax.AxIntegerMin()
' Output: -9223372036854775808

Set ax = Nothing
%>
```

71.8 API Reference

- **Object**: `G3AXON.FUNCTIONS`
- **Method**: `AxIntegerMin`
- **Arguments**: none
- **Returns**: `Integer` (minimum signed 64-bit integer value)

72 Round a Number Up

72.1 Overview

Returns the smallest integer value greater than or equal to the specified number (ceiling).

72.2 Prerequisites

Instantiate the library with `Server.CreateObject("G3AXON.FUNCTIONS")`.

72.3 Syntax

```
result = obj.AxCeil(number)
```

72.4 Parameters

Parameter	Type	Required	Description
number	Double	Yes	The numeric value to round up. Coerced to Double before processing.

72.5 Return Value

- **Double**: Returns the smallest integer value that is greater than or equal to `number`.
- **Integer**: Returns `0` when no argument is provided.

72.6 Remarks

- Method names are case-insensitive in VBScript dispatch.

72.7 Example

```
<%
Option Explicit
Dim ax
Set ax = Server.CreateObject("G3AXON.FUNCTIONS")

Response.Write ax.AxCeil(4.1)  ' Output: 5
Response.Write ax.AxCeil(4.9)  ' Output: 5
Response.Write ax.AxCeil(-4.1) ' Output: -4

Set ax = Nothing
%>
```

72.8 API Reference

- **Object**: `G3AXON.FUNCTIONS`
- **Method**: `AxCeil`
- **Arguments**: `number As Double`
- **Returns**: `Double` (ceiling of input)

73 Round a Number Down

73.1 Overview

Returns the largest integer value less than or equal to the specified number (floor).

73.2 Prerequisites

Instantiate the library with `Server.CreateObject("G3AXON.FUNCTIONS")`.

73.3 Syntax

```
result = obj.AxFloor(number)
```

73.4 Parameters

Parameter	Type	Required	Description
number	Double	Yes	The numeric value to round down. Coerced to Double before processing.

73.5 Return Value

- **Double**: Returns the largest integer value that is less than or equal to `number`.
- **Integer**: Returns `0` when no argument is provided.

73.6 Remarks

- Method names are case-insensitive in VBScript dispatch.

73.7 Example

```
<%
Option Explicit
Dim ax
Set ax = Server.CreateObject("G3AXON.FUNCTIONS")

Response.Write ax.AxFloor(4.9)  ' Output: 4
Response.Write ax.AxFloor(4.1)  ' Output: 4
Response.Write ax.AxFloor(-4.1) ' Output: -5

Set ax = Nothing
%>
```

73.8 API Reference

- **Object**: `G3AXON.FUNCTIONS`
- **Method**: `AxFloor`
- **Arguments**: `number As Double`
- **Returns**: `Double` (floor of input)

74 Generate a Random Integer

74.1 Overview

Returns a pseudo-random Integer. Supports three call forms: unconstrained, single-bound, and range-bound.

74.2 Prerequisites

Instantiate the library with `Server.CreateObject("G3AXON.FUNCTIONS")`.

74.3 Syntax

```
result = obj.AxRand([max])
result = obj.AxRand(min, max)
```

74.4 Parameters

Parameter	Type	Required	Description
(none)	—	Optional	When called with no arguments, returns any non-negative random Integer.
max	Integer	Optional	When called with one argument, returns an Integer between `0` and `max` (inclusive).
min	Integer	Optional	When called with two arguments, the lower bound (inclusive). Swapped automatically if greater than `max`.
max	Integer	Optional	When called with two arguments, the upper bound (inclusive).

74.5 Return Value

- **Integer**: Returns a pseudo-random Integer within the specified range (inclusive on both ends).

74.6 Remarks

- When `min` is greater than `max`, the bounds are automatically swapped.
- When `max` is `0` or less and only one argument is provided, returns `0`.
- Method names are case-insensitive in VBScript dispatch.

74.7 Example

```
<%
Option Explicit
Dim ax
Set ax = Server.CreateObject("G3AXON.FUNCTIONS")

' Any non-negative integer
Response.Write ax.AxRand() & "<br>"

' Between 0 and 10
Response.Write ax.AxRand(10) & "<br>"

' Between 1 and 100
Response.Write ax.AxRand(1, 100) & "<br>"

Set ax = Nothing
%>
```

74.8 API Reference

- **Object**: `G3AXON.FUNCTIONS`
- **Method**: `AxRand`
- **Arguments**: `[max As Integer]` or `[min As Integer, max As Integer]`
- **Returns**: `Integer` (pseudo-random integer within the specified range)

75 Format a Number with Grouping and Decimal Options

75.1 Overview

Formats a numeric value with configurable decimal places, decimal separator, and thousands separator.

75.2 Prerequisites

Instantiate the library with `Server.CreateObject("G3AXON.FUNCTIONS")`.

75.3 Syntax

```
result = obj.AxNumberFormat(number [, decimals [, decPoint [, thousandsSep]]])
```

75.4 Parameters

Parameter	Type	Required	Description
number	Double	Yes	The numeric value to format.
decimals	Integer	Optional	The number of

Parameter	Type	Required	Description
			decimal places to include. Defaults to 2.
decPoint	String	Optional	The decimal separator character. Defaults to ".".
thousandsSep	String	Optional	The thousands grouping separator. Defaults to ",". Pass an empty string to disable grouping.

75.5 Return Value

- **String**: Returns the formatted number as a string.
- **String**: Returns an empty string when no argument is provided.

75.6 Remarks

- Modeled after the PHP `number_format` function behavior.
- Method names are case-insensitive in VBScript dispatch.

75.7 Example

```
<%
Option Explicit
Dim ax
Set ax = Server.CreateObject("G3AXON.FUNCTIONS")

' US format: 1,234,567.89
Response.Write ax.AxNumberFormat(1234567.891, 2)

' European format: 1.234.567,89
Response.Write ax.AxNumberFormat(1234567.891, 2, ",", ".")

' No decimal, no grouping
Response.Write ax.AxNumberFormat(9876.5, 0, ".", "")
' Output: 9877

Set ax = Nothing
%>
```

75.8 API Reference

- **Object**: `G3AXON.FUNCTIONS`
- **Method**: `AxNumberFormat`
- **Arguments**: `number As Double [, decimals As Integer [, decPoint As String [, thousandsSep As String]]]`
- **Returns**: `String` (formatted number string)

76 Return the Constant Pi

76.1 Overview

Returns the mathematical constant Pi (π = 3.141592653589793…).

76.2 Prerequisites

Instantiate the library with `Server.CreateObject("G3AXON.FUNCTIONS")`.

76.3 Syntax

```
result = obj.AxPi()
```

76.4 Parameters

This method does not require parameters.

76.5 Return Value

- **Double**: Returns `3.141592653589793` (the IEEE-754 double-precision representation of Pi).

76.6 Remarks

- Method names are case-insensitive in VBScript dispatch.

76.7 Example

```
<%
Option Explicit
Dim ax, area, radius
Set ax = Server.CreateObject("G3AXON.FUNCTIONS")

radius = 5
area = ax.AxPi() * radius * radius
Response.Write "Circle area: " & area
' Output: Circle area: 78.5398163397448

Set ax = Nothing
%>
```

76.8 API Reference

- **Object**: `G3AXON.FUNCTIONS`
- **Method**: `AxPi`
- **Arguments**: none
- **Returns**: `Double` (3.141592653589793)

77 Get the Smallest Positive Float Value

77.1 Overview

Returns the smallest positive non-zero value representable by a 64-bit double-precision float.

77.2 Prerequisites

Instantiate the library with `Server.CreateObject("G3AXON.FUNCTIONS")`.

77.3 Syntax

```
result = obj.AxSmallestFloatValue()
```

77.4 Parameters

This method does not require parameters.

77.5 Return Value

- **Double**: Returns `5e-324` (equivalent to `math.SmallestNonzeroFloat64` in Go).

77.6 Remarks

- Use this value for epsilon comparisons to detect near-zero floating-point differences.
- Method names are case-insensitive in VBScript dispatch.

77.7 Example

```
<%
Option Explicit
Dim ax, tiny
Set ax = Server.CreateObject("G3AXON.FUNCTIONS")

tiny = ax.AxSmallestFloatValue()
Response.Write "Smallest float: " & tiny

Set ax = Nothing
%>
```

77.8 API Reference

- **Object**: `G3AXON.FUNCTIONS`
- **Method**: `AxSmallestFloatValue`
- **Arguments**: `none`
- **Returns**: `Double` (5e-324)

78 Get the Float Precision Digit Count

78.1 Overview

Returns the standard number of significant decimal digits for IEEE-754 double-precision floating-point values.

78.2 Prerequisites

Instantiate the library with `Server.CreateObject("G3AXON.FUNCTIONS")`.

78.3 Syntax

```
result = obj.AxFloatPrecisionDigits()
```

78.4 Parameters

This method does not require parameters.

78.5 Return Value

- **Integer**: Always returns `15`.

78.6 Remarks

- This is a constant reflecting IEEE-754 double-precision capability.
- Method names are case-insensitive in VBScript dispatch.

78.7 Example

```
<%
Option Explicit
Dim ax
Set ax = Server.CreateObject("G3AXON.FUNCTIONS")

Response.Write ax.AxFloatPrecisionDigits()
' Output: 15

Set ax = Nothing
%>
```

78.8 API Reference

- **Object**: `G3AXON.FUNCTIONS`
- **Method**: `AxFloatPrecisionDigits`
- **Arguments**: `none`
- **Returns**: `Integer` (always `15`)

79 Count Elements in an Array

79.1 Overview

Returns the total number of elements in a zero-based VBArray.

79.2 Prerequisites

Instantiate the library with `Server.CreateObject("G3AXON.FUNCTIONS")`.

79.3 Syntax

```
result = obj.AxCount(array)
```

79.4 Parameters

Parameter	Type	Required	Description
array	Array	Yes	The VBArray to measure.

79.5 Return Value

- **Integer**: Returns the number of elements in `array`.
- **Integer**: Returns `0` when the input is not a valid array or no argument is provided.

79.6 Remarks

- Equivalent to `UBound(arr) - LBound(arr) + 1` for single-dimensional arrays.
- Method names are case-insensitive in VBScript dispatch.

79.7 Example

```
<%
Option Explicit
Dim ax, items
Set ax = Server.CreateObject("G3AXON.FUNCTIONS")

items = Array("A", "B", "C", "D")
Response.Write ax.AxCount(items)
' Output: 4
```

```
Set ax = Nothing
%>
```

79.8 API Reference

- **Object**: `G3AXON.FUNCTIONS`
- **Method**: `AxCount`
- **Arguments**: `array As Array`
- **Returns**: `Integer` (number of elements)

80 Split a String into an Array

80.1 Overview

Splits a string by a specified delimiter and returns a zero-based VBArray of the resulting substrings.

80.2 Prerequisites

Instantiate the library with `Server.CreateObject("G3AXON.FUNCTIONS")`.

80.3 Syntax

```
arr = obj.AxExplode(delimiter, str [, limit])
```

80.4 Parameters

Parameter	Type	Required	Description
delimiter	String	Yes	The string used to split `str`. When empty, each character becomes a separate element.
str	String	Yes	The string to split.
limit	Integer	Optional	Maximum number of elements to return. When specified, the last element contains the remainder of the string.

80.5 Return Value

- **Array**: Returns a zero-based VBArray of String elements.

- **Array**: Returns an empty array when fewer than two arguments are provided.

80.6 Remarks

- When `delimiter` is empty, `str` is split into individual characters.
- When `limit` is greater than the actual number of parts, it has no effect.
- Method names are case-insensitive in VBScript dispatch.

80.7 Example

```
<%
Option Explicit
Dim ax, parts, i
Set ax = Server.CreateObject("G3AXON.FUNCTIONS")

parts = ax.AxExplode(",", "apple,banana,cherry")
For i = 0 To UBound(parts)
    Response.Write parts(i) & "<br>"
Next
' Output:
' apple
' banana
' cherry

' Limit to 2 parts
parts = ax.AxExplode(",", "a,b,c,d", 2)
Response.Write parts(0) & " / " & parts(1)
' Output: a / b

Set ax = Nothing
%>
```

80.8 API Reference

- **Object**: `G3AXON.FUNCTIONS`
- **Method**: `AxExplode`
- **Arguments**: `delimiter As String, str As String [, limit As Integer]`
- **Returns**: `Array` (zero-based VBArray of String elements)

81 Join Array Elements into a String

81.1 Overview

Joins all elements of a VBArray into a single string, separated by the specified glue string.

81.2 Prerequisites

Instantiate the library with `Server.CreateObject("G3AXON.FUNCTIONS")`.

81.3 Syntax

```
result = obj.AxImplode(glue, inputArray)
```

81.4 Parameters

Parameter	Type	Required	Description
glue	String	Yes	The string placed between each element of `inputArray`.
inputArray	Array	Yes	The zero-based VBArray whose elements are joined.

81.5 Return Value

- **String**: Returns a single string with each array element separated by `glue`.
- **String**: Returns an empty string when `inputArray` is not a valid array or fewer than two arguments are provided.

81.6 Remarks

- Method names are case-insensitive in VBScript dispatch.

81.7 Example

```
<%
Option Explicit
Dim ax, fruits, result
Set ax = Server.CreateObject("G3AXON.FUNCTIONS")

fruits = Array("apple", "banana", "cherry")
result = ax.AxImplode(", ", fruits)
Response.Write result
' Output: apple, banana, cherry

Set ax = Nothing
%>
```

81.8 API Reference

- **Object**: `G3AXON.FUNCTIONS`
- **Method**: `AxImplode`
- **Arguments**: `glue As String, inputArray As Array`
- **Returns**: `String` (joined string)

82 Reverse an Array

82.1 Overview

Returns a new array with the elements in reverse order. The original array is not modified.

82.2 Prerequisites

Instantiate the library with `Server.CreateObject("G3AXON.FUNCTIONS")`.

82.3 Syntax

```
reversed = obj.AxArrayReverse(inputArray)
```

82.4 Parameters

Parameter	Type	Required	Description
inputArray	Array	Yes	The source VBArray whose elements will be reversed.

82.5 Return Value

- **Array**: Returns a zero-based VBArray containing the same elements as `inputArray` in reverse order.
- **Array**: Returns an empty array when `inputArray` is not a valid array.

82.6 Remarks

- Method names are case-insensitive in VBScript dispatch.
- The function does not modify the original array in place; it returns a new array.

82.7 Example

```
<%
Option Explicit
Dim ax, colors, reversed, i
Set ax = Server.CreateObject("G3AXON.FUNCTIONS")

colors = Array("Red", "Green", "Blue")
reversed = ax.AxArrayReverse(colors)

For i = 0 To UBound(reversed)
    Response.Write reversed(i) & "<br>"
Next
' Output:
' Blue
' Green
```

```
' Red

Set ax = Nothing
%>
```

82.8 API Reference

- **Object**: `G3AXON.FUNCTIONS`
- **Method**: `AxArrayReverse`
- **Arguments**: `inputArray As Array`
- **Returns**: `Array` (reversed copy of the input array)

83 Create a Numeric Range Array

83.1 Overview

Builds and returns a zero-based VBArray containing Integer values from `start` to `end`, progressing by `step`.

83.2 Prerequisites

Instantiate the library with `Server.CreateObject("G3AXON.FUNCTIONS")`.

83.3 Syntax

```
arr = obj.AxRange(start, end [, step])
```

83.4 Parameters

Parameter	Type	Required	Description
start	Integer	Yes	The first value in the range.
end	Integer	Yes	The last value in the range (inclusive).
step	Integer	Optional	The increment between values. Defaults to `1`. A negative step generates a descending range.

83.5 Return Value

- **Array**: Returns a zero-based VBArray of Integer values from `start` to `end`.
- **Array**: Returns an empty array when fewer than two arguments are provided.

83.6 Remarks

- When `step` is `0`, it defaults to `1`.
- For a descending range, provide a negative `step` and set `start` greater than `end`.
- Method names are case-insensitive in VBScript dispatch.

83.7 Example

```
<%
Option Explicit
Dim ax, nums, i
Set ax = Server.CreateObject("G3AXON.FUNCTIONS")

' Ascending range 1 to 5
nums = ax.AxRange(1, 5)
For i = 0 To UBound(nums)
    Response.Write nums(i) & " "
Next
' Output: 1 2 3 4 5

' Step of 2
nums = ax.AxRange(0, 10, 2)
For i = 0 To UBound(nums)
    Response.Write nums(i) & " "
Next
' Output: 0 2 4 6 8 10

Set ax = Nothing
%>
```

83.8 API Reference

- **Object**: `G3AXON.FUNCTIONS`
- **Method**: `AxRange`
- **Arguments**: `start As Integer, end As Integer [, step As Integer]`
- **Returns**: `Array` (zero-based VBArray of Integer values)

84 Replace All Occurrences of a Substring

84.1 Overview

Replaces every occurrence of a search string within a source string and returns the modified result.

84.2 Prerequisites

Instantiate the library with `Server.CreateObject("G3AXON.FUNCTIONS")`.

84.3 Syntax

```
result = obj.AxStringReplace(search, replacement, subject)
```

84.4 Parameters

Parameter	Type	Required	Description
search	String	Yes	The substring to find in `subject`.
replacement	String	Yes	The string to substitute for each occurrence of `search`.
subject	String	Yes	The source string to search within.

84.5 Return Value

- **String**: Returns a copy of `subject` with all occurrences of `search` replaced by `replacement`.
- **String**: Returns an empty string when fewer than three arguments are provided.

84.6 Remarks

- The replacement is case-sensitive.
- All non-overlapping occurrences are replaced in a single pass.
- Method names are case-insensitive in VBScript dispatch.

84.7 Example

```
<%
Option Explicit
Dim ax, result
Set ax = Server.CreateObject("G3AXON.FUNCTIONS")

result = ax.AxStringReplace("world", "AxonASP", "Hello world")
Response.Write result
' Output: Hello AxonASP

' Remove a character
result = ax.AxStringReplace("-", "", "2025-04-18")
Response.Write result
' Output: 20250418

Set ax = Nothing
%>
```

84.8 API Reference

- **Object**: `G3AXON.FUNCTIONS`
- **Method**: `AxStringReplace`
- **Arguments**: `search As String, replacement As String, subject As String`
- **Returns**: `String` (modified copy of `subject`)

85 Pad a String to a Specified Length

85.1 Overview

Pads a string to a target length using a specified pad character or string. Supports right-padding, left-padding, and center-padding.

85.2 Prerequisites

Instantiate the library with `Server.CreateObject("G3AXON.FUNCTIONS")`.

85.3 Syntax

```
result = obj.AxPad(str, length [, padStr [, padType]])
```

85.4 Parameters

Parameter	Type	Required	Description
str	String	Yes	The source string to pad.
length	Integer	Yes	The target total length of the resulting string. When `str` is already this length or longer, `str` is returned unchanged.
padStr	String	Optional	The character or string to use for padding. Defaults to a single space.
padType	Integer	Optional	Padding direction: `0` = left, `1` = right (default), `2` = both sides (center).

85.5 Return Value

- **String**: Returns the padded string when `str` is shorter than `length`.
- **String**: Returns `str` unchanged when it is already equal to or longer than `length`.
- **String**: Returns an empty string when fewer than two arguments are provided.

85.6 Remarks

- When `padType` is `2` (both sides), the extra character is appended to the right side.
- Method names are case-insensitive in VBScript dispatch.

85.7 Example

```
<%
Option Explicit
Dim ax
Set ax = Server.CreateObject("G3AXON.FUNCTIONS")

' Right-pad with spaces (default)
Response.Write "[" & ax.AxPad("Hi", 8) & "]"           ' [Hi      ]

' Left-pad with zeros
Response.Write "[" & ax.AxPad("42", 6, "0", 0) & "]"   ' [000042]

' Center-pad with dashes
Response.Write "[" & ax.AxPad("Go", 8, "-", 2) & "]"   ' [---Go---]

Set ax = Nothing
%>
```

85.8 API Reference

- **Object**: `G3AXON.FUNCTIONS`
- **Method**: `AxPad`
- **Arguments**: `str As String, length As Integer [, padStr As String [, padType As Integer]]`
- **Returns**: `String` (padded string)

86 Repeat a String

86.1 Overview

Returns a new string formed by repeating the source string a specified number of times.

86.2 Prerequisites

Instantiate the library with `Server.CreateObject("G3AXON.FUNCTIONS")`.

86.3 Syntax

```
result = obj.AxRepeat(str, count)
```

86.4 Parameters

Parameter	Type	Required	Description
str	String	Yes	The string to repeat.
count	Integer	Yes	The number of times to repeat `str`. Negative values are treated as `0`.

86.5 Return Value

- **String**: Returns `str` concatenated `count` times.
- **String**: Returns an empty string when `count` is `0` or negative, or when fewer than two arguments are provided.

86.6 Remarks

- Method names are case-insensitive in VBScript dispatch.

86.7 Example

```
<%
Option Explicit
Dim ax
Set ax = Server.CreateObject("G3AXON.FUNCTIONS")

Response.Write ax.AxRepeat("AB", 4)
' Output: ABABABAB

Response.Write ax.AxRepeat("-", 20)
' Output: --------------------

Set ax = Nothing
%>
```

86.8 API Reference

- **Object**: `G3AXON.FUNCTIONS`
- **Method**: `AxRepeat`

- **Arguments**: `str As String, count As Integer`
- **Returns**: `String` (repeated string)

87 Uppercase the First Character

87.1 Overview

Returns a copy of the input string with the first Unicode character converted to uppercase.

87.2 Prerequisites

Instantiate the library with `Server.CreateObject("G3AXON.FUNCTIONS")`.

87.3 Syntax

```
result = obj.AxUcfirst(str)
```

87.4 Parameters

Parameter	Type	Required	Description
str	String	Yes	The source string.

87.5 Return Value

- **String**: Returns `str` with the first character converted to uppercase and remaining characters unchanged.
- **String**: Returns an empty string when `str` is empty or no argument is provided.

87.6 Remarks

- Method names are case-insensitive in VBScript dispatch.
- Only the first Unicode character is uppercased; all other characters are left as-is.

87.7 Example

```
<%
Option Explicit
Dim ax
Set ax = Server.CreateObject("G3AXON.FUNCTIONS")

Response.Write ax.AxUcfirst("hello world")
' Output: Hello world

Response.Write ax.AxUcfirst("axonASP")
' Output: AxonASP
```

```
Set ax = Nothing
%>
```

87.8 API Reference

- **Object**: `G3AXON.FUNCTIONS`
- **Method**: `AxUcfirst`
- **Arguments**: `str As String`
- **Returns**: `String` (string with uppercased first character)

88 Count Words in a String

88.1 Overview

Counts the number of words in a string, or optionally returns an array of those words.

88.2 Prerequisites

Instantiate the library with `Server.CreateObject("G3AXON.FUNCTIONS")`.

88.3 Syntax

```
result = obj.AxWordCount(str [, format])
```

88.4 Parameters

Parameter	Type	Required	Description
str	String	Yes	The string to analyze.
format	Integer	Optional	Output mode. `0` (default) returns the word count as an Integer. `1` returns a zero-based VBArray of String words.

88.5 Return Value

- **Integer**: Returns the number of whitespace-delimited words when `format` is `0` or omitted.
- **Array**: Returns a zero-based VBArray of String words when `format` is `1`.
- **Integer**: Returns `0` when no argument is provided.

88.6 Remarks

- Word boundaries are determined by whitespace (spaces, tabs, newlines).

- Method names are case-insensitive in VBScript dispatch.

88.7 Example

```
<%
Option Explicit
Dim ax, wordArr, i
Set ax = Server.CreateObject("G3AXON.FUNCTIONS")

' Count words
Response.Write ax.AxWordCount("The quick brown fox")
' Output: 4

' Get word array
wordArr = ax.AxWordCount("Hello World ASP", 1)
For i = 0 To UBound(wordArr)
    Response.Write wordArr(i) & "<br>"
Next
' Output:
' Hello
' World
' ASP

Set ax = Nothing
%>
```

88.8 API Reference

- **Object**: `G3AXON.FUNCTIONS`
- **Method**: `AxWordCount`
- **Arguments**: `str As String [, format As Integer]`
- **Returns**: `Integer` (word count) or `Array` (word list when format=1)

89 Convert Newlines to HTML Line Breaks

89.1 Overview

Replaces all newline sequences in a string with `<br>` HTML tags.

89.2 Prerequisites

Instantiate the library with `Server.CreateObject("G3AXON.FUNCTIONS")`.

89.3 Syntax

```
result = obj.AxNl2br(str)
```

89.4 Parameters

Parameter	Type	Required	Description

Parameter	Type	Required	Description
str	String	Yes	The source string containing newlines to replace.

89.5 Return Value

- **String**: Returns a copy of `str` with all `CRLF` (`\r\n`), `LF` (`\n`), and `CR` (`\r`) sequences replaced by `<br>`.
- **String**: Returns an empty string when no argument is provided.

89.6 Remarks

- `CRLF` sequences are replaced first before individual `LF` and `CR` characters.
- Method names are case-insensitive in VBScript dispatch.

89.7 Example

```
<%
Option Explicit
Dim ax, text
Set ax = Server.CreateObject("G3AXON.FUNCTIONS")

text = "Line 1" & vbCrLf & "Line 2" & vbCrLf & "Line 3"
Response.Write ax.AxNl2br(text)
' Output: Line 1<br>Line 2<br>Line 3

Set ax = Nothing
%>
```

89.8 API Reference

- **Object**: `G3AXON.FUNCTIONS`
- **Method**: `AxNl2br`
- **Arguments**: `str As String`
- **Returns**: `String` (string with newlines replaced by `<br>`)

90 Trim Characters from Both Ends of a String

90.1 Overview

Removes leading and trailing characters from a string. Defaults to stripping standard whitespace.

90.2 Prerequisites

Instantiate the library with `Server.CreateObject("G3AXON.FUNCTIONS")`.

90.3 Syntax

```
result = obj.AxTrim(str [, chars])
```

90.4 Parameters

Parameter	Type	Required	Description
str	String	Yes	The string to trim.
chars	String	Optional	A set of characters to strip from both ends. When omitted, the default set " \t\n\r\v\f" is used.

90.5 Return Value

- **String**: Returns `str` with all leading and trailing occurrences of any character in `chars` removed.
- **String**: Returns an empty string when no argument is provided.

90.6 Remarks

- `chars` works as a character set, not as a literal substring — any character in `chars` is trimmed.
- Method names are case-insensitive in VBScript dispatch.

90.7 Example

```
<%
Option Explicit
Dim ax
Set ax = Server.CreateObject("G3AXON.FUNCTIONS")

Response.Write ax.AxTrim("   Hello World   ")
' Output: Hello World

Response.Write ax.AxTrim("***data***", "*")
' Output: data

Set ax = Nothing
%>
```

90.8 API Reference

- **Object**: `G3AXON.FUNCTIONS`
- **Method**: `AxTrim`

- **Arguments**: `str As String [, chars As String]`
- **Returns**: `String` (trimmed string)

91 Parse a CSV Row String

91.1 Overview

Parses one CSV-formatted row string and returns its fields as a zero-based VBArray.

91.2 Prerequisites

Instantiate the library with `Server.CreateObject("G3AXON.FUNCTIONS")`.

91.3 Syntax

```
arr = obj.AxStringGetCSV(str [, delimiter])
```

91.4 Parameters

Parameter	Type	Required	Description
str	String	Yes	A single CSV row string to parse. Quoted fields and embedded delimiters are handled correctly.
delimiter	String	Optional	The field separator character. Only the first character in the string is used. Defaults to `,`.

91.5 Return Value

- **Array**: Returns a zero-based VBArray of String fields parsed from `str`.
- **Array**: Returns an empty array when no argument is provided or when parsing fails.

91.6 Remarks

- Only the first row of a multi-row CSV string is parsed.
- Quoted fields with embedded commas are handled correctly by the underlying CSV parser.
- Method names are case-insensitive in VBScript dispatch.

91.7 Example

```
<%
Option Explicit
Dim ax, fields
Set ax = Server.CreateObject("G3AXON.FUNCTIONS")

fields = ax.AxStringGetCSV("apple,\"banana,split\",cherry")
Response.Write fields(0) & "<br>" ' apple
Response.Write fields(1) & "<br>" ' banana,split
Response.Write fields(2) & "<br>" ' cherry

' Tab-delimited
fields = ax.AxStringGetCSV("col1" & Chr(9) & "col2", Chr(9))
Response.Write fields(0)          ' col1

Set ax = Nothing
%>
```

91.8 API Reference

- **Object**: `G3AXON.FUNCTIONS`
- **Method**: `AxStringGetCSV`
- **Arguments**: `str As String [, delimiter As String]`
- **Returns**: `Array` (zero-based VBArray of String fields)

92 Compute an MD5 Hash

92.1 Overview

Computes the MD5 hash of a string and returns it as a lowercase hexadecimal string.

92.2 Prerequisites

Instantiate the library with `Server.CreateObject("G3AXON.FUNCTIONS")`.

92.3 Syntax

```
result = ax.AxMD5(str)
```

92.4 Parameters

Parameter	Type	Required	Description
str	String	Yes	The input string to hash.

92.5 Return Value

- **String**: A 32-character lowercase hexadecimal MD5 digest.

92.6 Remarks

- MD5 is a fast, non-cryptographic-strength hash. Use it for checksums and non-security-sensitive identifiers, not for password storage.
- The input string is encoded as UTF-8 before hashing.
- Method names are case-insensitive.

92.7 Example

```
<%
Option Explicit
Dim ax, hash
Set ax = Server.CreateObject("G3AXON.FUNCTIONS")

hash = ax.AxMD5("AxonASP")
Response.Write hash
' Output: 3e3a3e7f84bbf16cce4ad33fd9d68aca (example — actual value depend
s on input)

hash = ax.AxMD5("admin")
Response.Write "<br>" & hash
' Output: 21232f297a57a5a743894a0e4a801fc3

Set ax = Nothing
%>
```

92.8 API Reference

- **Object**: `G3AXON.FUNCTIONS`
- **Method**: `AxMD5`
- **Arguments**: `str` (String, required)
- **Returns**: String — 32-character lowercase hexadecimal MD5 hash

93 Compute a SHA-1 Hash

93.1 Overview

Computes the SHA-1 hash of a string and returns it as a lowercase hexadecimal string.

93.2 Prerequisites

Instantiate the library with `Server.CreateObject("G3AXON.FUNCTIONS")`.

93.3 Syntax

```
result = ax.AxSHA1(str)
```

93.4 Parameters

Parameter	Type	Required	Description
str	String	Yes	The input string to hash.

93.5 Return Value

- **String**: A 40-character lowercase hexadecimal SHA-1 digest.

93.6 Remarks

- SHA-1 produces a 160-bit hash. It is suitable for checksums and integrity verification but is not recommended for new security-sensitive applications.
- The input string is encoded as UTF-8 before hashing.
- Method names are case-insensitive.

93.7 Example

```
<%
Option Explicit
Dim ax, hash
Set ax = Server.CreateObject("G3AXON.FUNCTIONS")

hash = ax.AxSHA1("test")
Response.Write hash
' Output: a94a8fe5ccb19ba61c4c0873d391e987982fbbd3

Set ax = Nothing
%>
```

93.8 API Reference

- **Object**: `G3AXON.FUNCTIONS`
- **Method**: `AxSHA1`
- **Arguments**: `str` (String, required)
- **Returns**: String — 40-character lowercase hexadecimal SHA-1 hash

94 Compute a String Hash by Algorithm

94.1 Overview

Computes the hash of a string using the specified cryptographic algorithm and returns the result as a lowercase hexadecimal string.

94.2 Prerequisites

Instantiate the library with `Server.CreateObject("G3AXON.FUNCTIONS")`.

94.3 Syntax

```
result = ax.AxHash(algo, str)
```

94.4 Parameters

Parameter	Type	Required	Description
algo	String	Yes	The hashing algorithm. Supported values: `"md5"`, `"sha1"`, `"sha256"`.
str	String	Yes	The input string to hash.

94.5 Return Value

- **String**: A lowercase hexadecimal digest. Length depends on the algorithm: 32 characters for MD5, 40 for SHA-1, 64 for SHA-256.
- **String (empty)**: Returned when an unsupported algorithm name is provided.

94.6 Remarks

- Algorithm names are matched case-insensitively.
- The input string is encoded as UTF-8 before hashing.
- Method names are case-insensitive.

94.7 Example

```
<%
Option Explicit
Dim ax
Set ax = Server.CreateObject("G3AXON.FUNCTIONS")

Response.Write ax.AxHash("md5", "hello") & "<br>"
' Output: 5d41402abc4b2a76b9719d911017c592

Response.Write ax.AxHash("sha256", "hello") & "<br>"
' Output: 2cf24dba5fb0a30e26e83b2ac5b9e29e1b161e5c1fa7425e73043362938b982
4

Response.Write ax.AxHash("invalid", "hello") & "<br>"
' Output: (empty string)

Set ax = Nothing
%>
```

94.8 API Reference

- **Object**: `G3AXON.FUNCTIONS`
- **Method**: `AxHash`
- **Arguments**: `algo` (String, required), `str` (String, required)
- **Returns**: String — lowercase hexadecimal hash, or empty string for unsupported algorithms

95 Encode a String to Base64

95.1 Overview

Encodes a string to Base64 format using the standard RFC 4648 encoding.

95.2 Prerequisites

Instantiate the library with `Server.CreateObject("G3AXON.FUNCTIONS")`.

95.3 Syntax

```
result = ax.AxBase64Encode(str)
```

95.4 Parameters

Parameter	Type	Required	Description
str	String	Yes	The string to encode.

95.5 Return Value

- **String**: The Base64-encoded representation of the input string.

95.6 Remarks

- The input string is encoded as UTF-8 before Base64 encoding.
- Base64 encoding is commonly used to safely transmit or store binary or text data in text-based protocols.
- To reverse the operation, use `AxBase64Decode`.
- Method names are case-insensitive.

95.7 Example

```
<%
Option Explicit
Dim ax, encoded
Set ax = Server.CreateObject("G3AXON.FUNCTIONS")

encoded = ax.AxBase64Encode("Hello World")
Response.Write encoded
' Output: SGVsbG8gV29ybGQ=
```

```
Set ax = Nothing
%>
```

95.8 API Reference

- **Object**: `G3AXON.FUNCTIONS`
- **Method**: `AxBase64Encode`
- **Arguments**: `str` (String, required)
- **Returns**: String — Base64-encoded representation of the input

96 Decode a Base64 String

96.1 Overview

Decodes a Base64-encoded string back to its original representation.

96.2 Prerequisites

Instantiate the library with `Server.CreateObject("G3AXON.FUNCTIONS")`.

96.3 Syntax

```
result = ax.AxBase64Decode(str)
```

96.4 Parameters

Parameter	Type	Required	Description
str	String	Yes	The Base64-encoded string to decode.

96.5 Return Value

- **String**: The decoded original string.
- **String (empty)**: Returned when the input is not valid Base64.

96.6 Remarks

- The decoded bytes are interpreted as a UTF-8 string.
- Padding characters (=) are handled automatically.
- To encode a string, use `AxBase64Encode`.
- Method names are case-insensitive.

96.7 Example

```
<%
Option Explicit
Dim ax, decoded
Set ax = Server.CreateObject("G3AXON.FUNCTIONS")
```

```
decoded = ax.AxBase64Decode("SGVsbG8gV29ybGQ=")
Response.Write decoded
' Output: Hello World

decoded = ax.AxBase64Decode("!!!invalid!!!")
Response.Write "<br>" & Len(decoded)
' Output: 0 (empty string returned for invalid input)

Set ax = Nothing
%>
```

96.8 API Reference

- **Object**: `G3AXON.FUNCTIONS`
- **Method**: `AxBase64Decode`
- **Arguments**: `str` (String, required)
- **Returns**: String — decoded original string, or empty string on invalid Base64 input

97 Decode a URL-Encoded String

97.1 Overview

Decodes a percent-encoded URL string produced by standard form submission or query string encoding.

97.2 Prerequisites

Instantiate the library with `Server.CreateObject("G3AXON.FUNCTIONS")`.

97.3 Syntax

```
result = ax.AxUrlDecode(str)
```

97.4 Parameters

Parameter	Type	Required	Description
str	String	Yes	The URL-encoded string to decode.

97.5 Return Value

- **String**: The decoded string.
- **String (original input)**: Returned unchanged when the input contains malformed percent-encoding sequences.

97.6 Remarks

- This method uses standard RFC 3986 percent-decoding. It does **not** convert + to a space. Use `AxRawUrlDecode` if the input uses + as a space character.

- Method names are case-insensitive.

97.7 Example

```
<%
Option Explicit
Dim ax, decoded
Set ax = Server.CreateObject("G3AXON.FUNCTIONS")

decoded = ax.AxUrlDecode("Hello%20World%21")
Response.Write decoded
' Output: Hello World!

decoded = ax.AxUrlDecode("Hello+World")
Response.Write "<br>" & decoded
' Output: Hello+World (+ is not converted — use AxRawUrlDecode for that)

Set ax = Nothing
%>
```

97.8 API Reference

- **Object**: `G3AXON.FUNCTIONS`
- **Method**: `AxUrlDecode`
- **Arguments**: `str` (String, required)
- **Returns**: String — decoded string, or original input on malformed encoding

98 Decode a Raw URL-Encoded String

98.1 Overview

Decodes a URL-encoded string, treating + characters as spaces before applying standard percent-decoding.

98.2 Prerequisites

Instantiate the library with `Server.CreateObject("G3AXON.FUNCTIONS")`.

98.3 Syntax

```
result = ax.AxRawUrlDecode(str)
```

98.4 Parameters

Parameter	Type	Required	Description
str	String	Yes	The raw URL-encoded string to decode.

98.5 Return Value

- **String**: The decoded string, with + characters replaced by spaces and percent-encoded sequences unescaped.

98.6 Remarks

- This method first replaces each + with a space, then applies RFC 3986 percent-decoding. This matches the behavior of HTML form submissions using `application/x-www-form-urlencoded` encoding.
- Use `AxUrlDecode` when the input does not use + as a space.
- Method names are case-insensitive.

98.7 Example

```
<%
Option Explicit
Dim ax, decoded
Set ax = Server.CreateObject("G3AXON.FUNCTIONS")

decoded = ax.AxRawUrlDecode("Hello+World%21")
Response.Write decoded
' Output: Hello World!

Set ax = Nothing
%>
```

98.8 API Reference

- **Object**: `G3AXON.FUNCTIONS`
- **Method**: `AxRawUrlDecode`
- **Arguments**: `str` (String, required)
- **Returns**: String — decoded string with + converted to spaces

99 Convert RGB Values to a Hexadecimal Color Code

99.1 Overview

Converts individual red, green, and blue color component values to an uppercase HTML hexadecimal color string.

99.2 Prerequisites

Instantiate the library with `Server.CreateObject("G3AXON.FUNCTIONS")`.

99.3 Syntax

```
result = ax.AxRgbToHex(r, g, b)
```

99.4 Parameters

Parameter	Type	Required	Description
r	Integer	Yes	Red component (0–255). Values outside this range are masked with `& 0xFF`.
g	Integer	Yes	Green component (0–255). Values outside this range are masked with `& 0xFF`.
b	Integer	Yes	Blue component (0–255). Values outside this range are masked with `& 0xFF`.

99.5 Return Value

- **String**: An uppercase hexadecimal color string in `#RRGGBB` format (e.g., `"#FF8000"`).
- **String `"#000000"`**: Returned when fewer than three arguments are supplied.

99.6 Remarks

- Each component is masked with `& 0xFF` before formatting, so values outside 0–255 wrap rather than cause an error.
- The hex digits are uppercase (A–F).
- Method names are case-insensitive.

99.7 Example

```
<%
Option Explicit
Dim ax, hex
Set ax = Server.CreateObject("G3AXON.FUNCTIONS")

hex = ax.AxRgbToHex(255, 128, 0)
Response.Write hex
' Output: #FF8000

hex = ax.AxRgbToHex(0, 0, 255)
Response.Write "<br>" & hex
' Output: #0000FF
```

```
Set ax = Nothing
%>
```

99.8 API Reference

- **Object**: `G3AXON.FUNCTIONS`
- **Method**: `AxRgbToHex`
- **Arguments**: `r` (Integer, required), `g` (Integer, required), `b` (Integer, required)
- **Returns**: String — uppercase `#RRGGBB` hex color code

100 Convert a Hexadecimal Color Code to RGB

100.1 Overview

Converts an HTML hexadecimal color string to a CSS `rgb()` function string.

100.2 Prerequisites

Instantiate the library with `Server.CreateObject("G3AXON.FUNCTIONS")`.

100.3 Syntax

```
result = ax.AxHexToRgb(hex)
```

100.4 Parameters

Parameter	Type	Required	Description
hex	String	Yes	An HTML color string in `#RGB` or `#RRGGBB` format. The leading `#` is required.

100.5 Return Value

- **String**: A CSS color string in `rgb(R,G,B)` format (e.g., `"rgb(0,128,255)"`).
- **String `"rgb(0,0,0)"`**: Returned when the input is empty, missing the `#` prefix, or has an invalid format.

100.6 Remarks

- Both 3-digit (`#RGB`) and 6-digit (`#RRGGBB`) formats are accepted.
- The 3-digit form is expanded by doubling each digit (e.g., `#F80` ⇢ `#FF8800`).
- Method names are case-insensitive.

100.7 Example

```
<%
Option Explicit
Dim ax, rgb
Set ax = Server.CreateObject("G3AXON.FUNCTIONS")

rgb = ax.AxHexToRgb("#0080FF")
Response.Write rgb
' Output: rgb(0,128,255)

rgb = ax.AxHexToRgb("#F80")
Response.Write "<br>" & rgb
' Output: rgb(255,136,0)

rgb = ax.AxHexToRgb("invalid")
Response.Write "<br>" & rgb
' Output: rgb(0,0,0)

Set ax = Nothing
%>
```

100.8 API Reference

- **Object**: `G3AXON.FUNCTIONS`
- **Method**: `AxHexToRgb`
- **Arguments**: `hex` (String, required)
- **Returns**: String — `rgb(R,G,B)` form, or `rgb(0,0,0)` on invalid input

101 Get the Default Logo as a Base64 Data URI

101.1 Overview

Returns the server's configured default logo image as an inline Base64 data URI string, ready for use in an HTML `<img>` tag.

101.2 Prerequisites

Instantiate the library with `Server.CreateObject("G3AXON.FUNCTIONS")`.

Set the logo file path in `config/axonasp.toml`:

```
[axfunctions]
ax_default_logo_path = "./www/images/logo.png"
```

101.3 Syntax

```
result = ax.AxGetLogo()
```

101.4 Parameters

This method does not accept any parameters.

101.5 Return Value

- **String**: A data URI in the form `data:{mime}; base64,{base64content}` where `{mime}` is the detected MIME type (e.g., `image/png`) and `{base64content}` is the Base64-encoded file content.
- **String (empty)**: Returned when `ax_default_logo_path` is not configured or the file cannot be read.

101.6 Remarks

- The MIME type is detected from the file extension. Supported extensions include `.png`, `.jpg`, `.gif`, `.svg`, and `.ico`.
- The file path is resolved relative to the AxonASP executable directory.
- Method names are case-insensitive.

101.7 Example

```
<%
Option Explicit
Dim ax, logoUri
Set ax = Server.CreateObject("G3AXON.FUNCTIONS")

logoUri = ax.AxGetLogo()

If logoUri <> "" Then
    Response.Write "<img src=""" & logoUri & """ alt=""Logo"">"
End If

Set ax = Nothing
%>
```

101.8 API Reference

- **Object**: `G3AXON.FUNCTIONS`
- **Method**: `AxGetLogo`
- **Arguments**: None
- **Returns**: String — `data:{mime}; base64,{content}` URI, or empty string if not configured or unreadable

102 Escape HTML Special Characters

102.1 Overview

Escapes the HTML special characters in a string to their corresponding HTML entities, preventing browsers from interpreting them as markup.

102.2 Prerequisites

Instantiate the library with `Server.CreateObject("G3AXON.FUNCTIONS")`.

102.3 Syntax

```
result = ax.AxHtmlSpecialChars(str)
```

102.4 Parameters

Parameter	Type	Required	Description
str	String	Yes	The string to escape.

102.5 Return Value

- **String**: The input string with &, <, >, and " replaced by their HTML entity equivalents.
- **String (empty)**: Returned when no argument is supplied.

102.6 Remarks

- Characters replaced: & ⟶ `&`, < ⟶ `<`, > ⟶ `>`, " ⟶ `"`.
- Use this method when outputting user-supplied data into HTML to prevent cross-site scripting (XSS) attacks.
- Method names are case-insensitive.

102.7 Example

```
<%
Option Explicit
Dim ax, safe
Set ax = Server.CreateObject("G3AXON.FUNCTIONS")

safe = ax.AxHtmlSpecialChars("<script>alert('xss')</script>")
Response.Write safe
' Output: &lt;script&gt;alert(&#34;xss&#34;)&lt;/script&gt;

Set ax = Nothing
%>
```

102.8 API Reference

- **Object**: `G3AXON.FUNCTIONS`

- **Method**: `AxHtmlSpecialChars`
- **Arguments**: `str` (String, required)
- **Returns**: String — HTML-entity-escaped version of the input

103 Strip HTML and XML Tags from a String

103.1 Overview

Removes all HTML and XML tags from a string, returning only the plain text content.

103.2 Prerequisites

Instantiate the library with `Server.CreateObject("G3AXON.FUNCTIONS")`.

103.3 Syntax

```
result = ax.AxStripTags(str)
```

103.4 Parameters

Parameter	Type	Required	Description
str	String	Yes	The string from which to remove HTML and XML tags.

103.5 Return Value

- **String**: The input string with all content between < and > removed.
- **String (empty)**: Returned when no argument is supplied.

103.6 Remarks

- Tags are identified and removed using a regular expression that matches any sequence from < to >.
- Comments (`<!-- ... -->`) and processing instructions (`<? ... ?>`) are also removed.
- This method does not decode HTML entities. Use `AxHtmlSpecialChars` to escape output, not sanitize HTML.
- Method names are case-insensitive.

103.7 Example

```
<%
Option Explicit
Dim ax, html, plain
Set ax = Server.CreateObject("G3AXON.FUNCTIONS")
```

```
html = "<p>Hello <b>AxonASP</b>!</p><!-- comment -->"
plain = ax.AxStripTags(html)
Response.Write plain
' Output: Hello AxonASP!

Set ax = Nothing
%>
```

103.8 API Reference

- **Object**: `G3AXON.FUNCTIONS`
- **Method**: `AxStripTags`
- **Arguments**: `str` (String, required)
- **Returns**: String — input with all HTML/XML tags removed

104 Validate an IP Address

104.1 Overview

Determines whether a string is a valid IPv4 or IPv6 address.

104.2 Prerequisites

Instantiate the library with `Server.CreateObject("G3AXON.FUNCTIONS")`.

104.3 Syntax

```
result = ax.AxFilterValidateIp(ipAddress)
```

104.4 Parameters

Parameter	Type	Required	Description
ipAddress	String	Yes	The string to validate as an IP address.

104.5 Return Value

- **Boolean `True`**: The string is a valid IPv4 or IPv6 address.
- **Boolean `False`**: The string is not a valid IP address, or no argument was supplied.

104.6 Remarks

- Validation uses Go's `net.ParseIP`, which accepts both IPv4 (e.g., `192.168.1.1`) and IPv6 (e.g., `::1`) addresses.
- This method does not perform DNS resolution or reachability checks.
- Method names are case-insensitive.

104.7 Example

```
<%
Option Explicit
Dim ax
Set ax = Server.CreateObject("G3AXON.FUNCTIONS")

If ax.AxFilterValidateIp("192.168.1.1") Then
    Response.Write "192.168.1.1 is valid.<br>"
End If

If ax.AxFilterValidateIp("::1") Then
    Response.Write "::1 is a valid IPv6 address.<br>"
End If

If Not ax.AxFilterValidateIp("not-an-ip") Then
    Response.Write "not-an-ip is invalid.<br>"
End If

Set ax = Nothing
%>
```

104.8 API Reference

- **Object**: `G3AXON.FUNCTIONS`
- **Method**: `AxFilterValidateIp`
- **Arguments**: `ipAddress` (String, required)
- **Returns**: Boolean — `True` if valid IPv4 or IPv6; `False` otherwise

105 Validate an Email Address

105.1 Overview

Determines whether a string is a syntactically valid email address.

105.2 Prerequisites

Instantiate the library with `Server.CreateObject("G3AXON.FUNCTIONS")`.

105.3 Syntax

```
result = ax.AxFilterValidateEmail(emailAddress)
```

105.4 Parameters

Parameter	Type	Required	Description
emailAddress	String	Yes	The string to validate as an

Parameter	Type	Required	Description
			email address.

105.5 Return Value

- **Boolean `True`**: The string is a syntactically valid email address.
- **Boolean `False`**: The string is not a valid email address, or no argument was supplied.

105.6 Remarks

- Validation uses Go's `mail.ParseAddress`, which checks RFC 5322 syntax. It does not perform DNS MX record lookups or verify the mailbox exists.
- Addresses with display names (e.g., `John Doe <john@example.com>`) are accepted.
- Method names are case-insensitive.

105.7 Example

```
<%
Option Explicit
Dim ax
Set ax = Server.CreateObject("G3AXON.FUNCTIONS")

If ax.AxFilterValidateEmail("user@example.com") Then
    Response.Write "user@example.com is valid.<br>"
End If

If Not ax.AxFilterValidateEmail("not-an-email") Then
    Response.Write "not-an-email is invalid.<br>"
End If

Set ax = Nothing
%>
```

105.8 API Reference

- **Object**: `G3AXON.FUNCTIONS`
- **Method**: `AxFilterValidateEmail`
- **Arguments**: `emailAddress` (String, required)
- **Returns**: Boolean — `True` if syntactically valid email; `False` otherwise

106 Check If a Value Is an Integer

106.1 Overview

Determines whether the VM internal type of a value is `VTInteger` (Integer).

106.2 Prerequisites

Instantiate the library with `Server.CreateObject("G3AXON.FUNCTIONS")`.

106.3 Syntax

```
result = ax.AxIsInt(value)
```

106.4 Parameters

Parameter	Type	Required	Description
value	Variant	Yes	The value to inspect.

106.5 Return Value

- **Boolean `True`**: The value's VM internal type is `VTInteger`.
- **Boolean `False`**: The value is not a `VTInteger`, or no argument was supplied.

106.6 Remarks

- This method checks the VM's internal type tag, not the VBScript `VarType`. A value returned from a function that coerces to an integer at the VBScript level may still fail this check if the VM stored it as a different type.
- Use `AxIsFloat` to check for Double precision values.
- Method names are case-insensitive.

106.7 Example

```
<%
Option Explicit
Dim ax, a, b
Set ax = Server.CreateObject("G3AXON.FUNCTIONS")

a = 100
b = 100.5

Response.Write ax.AxIsInt(a) & "<br>"   ' True
Response.Write ax.AxIsInt(b) & "<br>"   ' False

Set ax = Nothing
%>
```

106.8 API Reference

- **Object**: `G3AXON.FUNCTIONS`
- **Method**: `AxIsInt`
- **Arguments**: `value` (Variant, required)
- **Returns**: Boolean — `True` if VM type is `VTInteger`; `False` otherwise

107 Check If a Value Is a Floating-Point Number

107.1 Overview

Determines whether the VM internal type of a value is `VTDouble` (Double precision floating-point).

107.2 Prerequisites

Instantiate the library with `Server.CreateObject("G3AXON.FUNCTIONS")`.

107.3 Syntax

```
result = ax.AxIsFloat(value)
```

107.4 Parameters

Parameter	Type	Required	Description
value	Variant	Yes	The value to inspect.

107.5 Return Value

- **Boolean `True`**: The value's VM internal type is `VTDouble`.
- **Boolean `False`**: The value is not a `VTDouble`, or no argument was supplied.

107.6 Remarks

- This method checks the VM's internal type tag, not the VBScript `VarType`.
- Use `AxIsInt` to check for Integer values.
- Method names are case-insensitive.

107.7 Example

```
<%
Option Explicit
Dim ax, a, b
Set ax = Server.CreateObject("G3AXON.FUNCTIONS")

a = 3.14
b = 3

Response.Write ax.AxIsFloat(a) & "<br>"   ' True
Response.Write ax.AxIsFloat(b) & "<br>"   ' False

Set ax = Nothing
%>
```

107.8 API Reference

- **Object**: `G3AXON.FUNCTIONS`

- **Method**: `AxIsFloat`
- **Arguments**: `value` (Variant, required)
- **Returns**: Boolean — `True` if VM type is `VTDouble`; `False` otherwise

108 Check If a String Contains Only Alphabetic Characters

108.1 Overview

Determines whether every character in a string is an ASCII alphabetic letter (a–z or A–Z).

108.2 Prerequisites

Instantiate the library with `Server.CreateObject("G3AXON.FUNCTIONS")`.

108.3 Syntax

```
result = ax.AxCtypeAlpha(str)
```

108.4 Parameters

Parameter	Type	Required	Description
str	String	Yes	The string to test.

108.5 Return Value

- **Boolean `True`**: All characters in the string are ASCII alphabetic letters.
- **Boolean `False`**: The string contains non-alphabetic characters, the string is empty, or no argument was supplied.

108.6 Remarks

- Only ASCII characters `A`–`Z` and `a`–`z` are considered alphabetic. Accented characters and Unicode letters return `False`.
- An empty string always returns `False`.
- Method names are case-insensitive.

108.7 Example

```
<%
Option Explicit
Dim ax
Set ax = Server.CreateObject("G3AXON.FUNCTIONS")

Response.Write ax.AxCtypeAlpha("AxonASP") & "<br>"    ' True
Response.Write ax.AxCtypeAlpha("Axon2") & "<br>"     ' False (digit)
Response.Write ax.AxCtypeAlpha("") & "<br>"          ' False (empty)
```

```
Set ax = Nothing
%>
```

108.8 API Reference

- **Object**: `G3AXON.FUNCTIONS`
- **Method**: `AxCtypeAlpha`
- **Arguments**: `str` (String, required)
- **Returns**: Boolean — `True` if all characters are ASCII alphabetic; `False` otherwise

109 Check If a String Contains Only Alphanumeric Characters

109.1 Overview

Determines whether every character in a string is an ASCII alphanumeric character (a–z, A–Z, or 0–9).

109.2 Prerequisites

Instantiate the library with `Server.CreateObject("G3AXON.FUNCTIONS")`.

109.3 Syntax

```
result = ax.AxCtypeAlnum(str)
```

109.4 Parameters

Parameter	Type	Required	Description
str	String	Yes	The string to test.

109.5 Return Value

- **Boolean `True`**: All characters in the string are ASCII alphanumeric.
- **Boolean `False`**: The string contains non-alphanumeric characters, the string is empty, or no argument was supplied.

109.6 Remarks

- Only ASCII letters `A–Z`, `a–z`, and digits `0–9` are considered alphanumeric. Spaces, punctuation, and Unicode characters return `False`.
- An empty string always returns `False`.
- Method names are case-insensitive.

109.7 Example

```
<%
Option Explicit
Dim ax
```

```
Set ax = Server.CreateObject("G3AXON.FUNCTIONS")

Response.Write ax.AxCtypeAlnum("AxonASP2026") & "<br>"   ' True
Response.Write ax.AxCtypeAlnum("Axon-ASP") & "<br>"     ' False (hyphen)
Response.Write ax.AxCtypeAlnum("") & "<br>"             ' False (empty)

Set ax = Nothing
%>
```

109.8 API Reference

- **Object**: `G3AXON.FUNCTIONS`
- **Method**: `AxCtypeAlnum`
- **Arguments**: `str` (String, required)
- **Returns**: Boolean — `True` if all characters are ASCII alphanumeric; `False` otherwise

110 Check If a Value Is Empty

110.1 Overview

Determines whether a value is considered empty under extended criteria that covers uninitialized variables, null values, zero-like values, and empty strings.

110.2 Prerequisites

Instantiate the library with `Server.CreateObject("G3AXON.FUNCTIONS")`.

110.3 Syntax

```
result = ax.AxEmpty(value)
```

110.4 Parameters

Parameter	Type	Required	Description
value	Variant	Yes	The value to test for emptiness.

110.5 Return Value

- **Boolean `True`**: The value is `Empty`, `Null`, an empty string (`""`), Integer `0`, Double `0.0`, or Boolean `False`.
- **Boolean `False`**: The value is any other non-empty, non-zero value.

110.6 Remarks

- This method provides a single call to replace multiple `IsEmpty`, `IsNull`, and zero-comparison checks.
- Use `AxIsSet` for the inverse check (value is initialized and not null).

- Method names are case-insensitive.

110.7 Example

```
<%
Option Explicit
Dim ax
Set ax = Server.CreateObject("G3AXON.FUNCTIONS")

Response.Write ax.AxEmpty("") & "<br>"          ' True
Response.Write ax.AxEmpty(0) & "<br>"           ' True
Response.Write ax.AxEmpty(False) & "<br>"       ' True
Response.Write ax.AxEmpty("hello") & "<br>"     ' False
Response.Write ax.AxEmpty(1) & "<br>"           ' False

Set ax = Nothing
%>
```

110.8 API Reference

- **Object**: `G3AXON.FUNCTIONS`
- **Method**: `AxEmpty`
- **Arguments**: `value` (Variant, required)
- **Returns**: Boolean — `True` if empty/null/zero-like; `False` otherwise End If

Set ax = Nothing %›

```
# Check If a Value Is Set

## Overview

Determines whether a value has been initialized and is neither `Empty` no
r `Null`.

## Prerequisites

Instantiate the library with `Server.CreateObject("G3AXON.FUNCTIONS")`.

## Syntax

```asp
result = ax.AxIsSet(value)
```

110.9 Parameters

Parameter	Type	Required	Description
value	Variant	Yes	The value to inspect.
```

110.10 Return Value

- **Boolean `True`:** The value is initialized and is not `Empty` or `Null`.
- **Boolean `False`:** The value is `Empty` or `Null`.

110.11 Remarks

- This method is the inverse of the VBScript `IsEmpty` and `IsNull` checks combined.
- Unlike `AxEmpty`, this method does not consider zero (`0`), empty string (`""`), or `False` as unset. Only `Empty` and `Null` return `False`.
- Method names are case-insensitive.

110.12 Example

```
<%
Option Explicit
Dim ax, val
Set ax = Server.CreateObject("G3AXON.FUNCTIONS")

' val is uninitialized (Empty)
Response.Write ax.AxIsSet(val) & "<br>"   ' False

val = 0
Response.Write ax.AxIsSet(val) & "<br>"   ' True (0 is a set value)

val = "hello"
Response.Write ax.AxIsSet(val) & "<br>"   ' True

Set ax = Nothing
%>
```

110.13 API Reference

- **Object:** `G3AXON.FUNCTIONS`
- **Method:** `AxIsSet`
- **Arguments:** `value` (Variant, required)
- **Returns:** Boolean — `True` if value is not `Empty` or `Null`; `False` otherwise

111 Get the Current Unix Timestamp

111.1 Overview

Returns the number of seconds elapsed since 1970-01-01 00:00:00 UTC (Unix epoch).

111.2 Prerequisites

Instantiate the library with `Server.CreateObject("G3AXON.FUNCTIONS")`.

111.3 Syntax

```
result = ax.AxTime()
```

111.4 Parameters

This method does not accept any parameters.

111.5 Return Value

- **Integer**: The current Unix timestamp in seconds.

111.6 Remarks

- The returned value reflects the server's clock in UTC, adjusted for the configured time zone.
- Pass the return value to `AxDate` to format it as a readable date/time string.
- Method names are case-insensitive.

111.7 Example

```
<%
Option Explicit
Dim ax, ts
Set ax = Server.CreateObject("G3AXON.FUNCTIONS")

ts = ax.AxTime()
Response.Write "Unix timestamp: " & ts & "<br>"
Response.Write "Formatted: " & ax.AxDate("Y-m-d H:i:s", ts)

Set ax = Nothing
%>
```

111.8 API Reference

- **Object**: `G3AXON.FUNCTIONS`
- **Method**: `AxTime`
- **Arguments**: None
- **Returns**: Integer — current Unix timestamp in seconds

112 Format a Date and Time String

112.1 Overview

Formats a Unix timestamp (or the current server time if omitted) into a readable string using PHP-compatible format tokens.

112.2 Prerequisites

Instantiate the library with `Server.CreateObject("G3AXON.FUNCTIONS")`.

112.3 Syntax

```
result = ax.AxDate(format [, timestamp])
```

112.4 Parameters

Parameter	Type	Required	Description
format	String	Yes	A format string composed of PHP-like tokens.
timestamp	Integer	No	A Unix timestamp (seconds since 1970-01-01 UTC). Defaults to the current server time when omitted.

Supported format tokens:

Token	Description	Example
`Y`	4-digit year	`2025`
`m`	Month with leading zero	`01`–`12`
`d`	Day of month with leading zero	`01`–`31`
`H`	Hour in 24-hour format with leading zero	`00`–`23`
`i`	Minutes with leading zero	`00`–`59`
`s`	Seconds with leading zero	`00`–`59`
`a`	Lowercase ante/post meridiem	`am` / `pm`
`A`	Uppercase ante/post meridiem	`AM` / `PM`
`F`	Full month name	`January`
`M`	Abbreviated month name	`Jan`
`l`	Full weekday name	`Monday`
`D`	Abbreviated weekday name	`Mon`
`N`	ISO-8601 day of the week (1=Monday, 7=Sunday)	`1`–`7`
`j`	Day of month without leading zero	`1`–`31`
`n`	Month without leading zero	`1`–`12`
`G`	Hour in 24-hour format without leading zero	`0`–`23`
`U`	Unix timestamp	Integer

112.5 Return Value

- **String**: The input format string with all recognized tokens replaced by their corresponding date/time values.

112.6 Remarks

- Month and weekday names use the locale configured for the AxonASP VM.
- Unrecognized characters in the format string are passed through unchanged.
- Method names are case-insensitive.

112.7 Example

```
<%
Option Explicit
Dim ax
Set ax = Server.CreateObject("G3AXON.FUNCTIONS")

Response.Write ax.AxDate("Y-m-d H:i:s") & "<br>"
' Output (example): 2025-06-15 14:30:00

Response.Write ax.AxDate("l, F j, Y", 1735689600) & "<br>"
' Output (example): Wednesday, January 1, 2025

Set ax = Nothing
%>
```

112.8 API Reference

- **Object**: `G3AXON.FUNCTIONS`
- **Method**: `AxDate`
- **Arguments**: `format` (String, required), `timestamp` (Integer, optional)
- **Returns**: String — formatted date/time string

113 Get the Last Modified Timestamp of the Current Page

113.1 Overview

Returns the Unix timestamp of the last modification time of the currently executing ASP script file.

113.2 Prerequisites

Instantiate the library with `Server.CreateObject("G3AXON.FUNCTIONS")`.

113.3 Syntax

```
result = ax.AxLastModified()
```

113.4 Parameters

This method does not accept any parameters.

113.5 Return Value

- **Integer**: Unix timestamp of the current file's last modification time.
- **Integer `0`**: Returned when the modification time cannot be determined (for example, when the script path is unavailable).

113.6 Remarks

- The method resolves the current script path using `Server.MapPath("")` internally.
- Use the return value with `AxDate` to format it as a human-readable string.
- Method names are case-insensitive.

113.7 Example

```
<%
Option Explicit
Dim ax, ts
Set ax = Server.CreateObject("G3AXON.FUNCTIONS")

ts = ax.AxLastModified()

If ts > 0 Then
    Response.Write "Page last modified: " & ax.AxDate("Y-m-d H:i:s", ts)
Else
    Response.Write "Modification time unavailable."
End If

Set ax = Nothing
%>
```

113.8 API Reference

- **Object**: `G3AXON.FUNCTIONS`
- **Method**: `AxLastModified`
- **Arguments**: None
- **Returns**: Integer — Unix timestamp of the script's last modification, or `0` on failure

114 Fetch the Content of a Remote URL

114.1 Overview

Performs a synchronous HTTP GET request to a URL and returns the response body as a string.

114.2 Prerequisites

Instantiate the library with `Server.CreateObject("G3AXON.FUNCTIONS")`.

114.3 Syntax

```
result = ax.AxGetRemoteFile(url)
```

114.4 Parameters

Parameter	Type	Required	Description
url	String	Yes	The fully qualified URL to request. Must begin with `http://` or `https://`.

114.5 Return Value

- **String**: The response body when the server returns HTTP 200 OK.
- **Boolean `False`**: Returned when the URL does not start with `http://` or `https://`, the request times out, a network error occurs, or the server returns a non-200 status code.

114.6 Remarks

- The default request timeout is 10 seconds.
- No request headers, authentication, or POST body are supported. For advanced HTTP scenarios, use the `G3HTTP` library.
- Check the return type with `VarType` or `TypeName` before using the result to handle failures gracefully.
- Method names are case-insensitive.

114.7 Example

```
<%
Option Explicit
Dim ax, content
Set ax = Server.CreateObject("G3AXON.FUNCTIONS")

content = ax.AxGetRemoteFile("https://example.com/api/data.json")
```

```
If VarType(content) = vbString Then
    Response.Write Server.HTMLEncode(content)
Else
    Response.Write "Failed to fetch the remote file."
End If

Set ax = Nothing
%>
```

114.8 API Reference

- **Object**: `G3AXON.FUNCTIONS`
- **Method**: `AxGetRemoteFile`
- **Arguments**: `url` (String, required)
- **Returns**: String (response body on HTTP 200) or Boolean `False` on failure

115 Generate a GUID

115.1 Overview

Generates a cryptographically secure version 4 GUID (Globally Unique Identifier) string.

115.2 Prerequisites

Instantiate the library with `Server.CreateObject("G3AXON.FUNCTIONS")`.

115.3 Syntax

```
result = ax.AxGenerateGuid()
```

115.4 Parameters

This method does not accept any parameters.

115.5 Return Value

- **String**: A 36-character RFC 4122 version 4 GUID in the form `xxxxxxxx-xxxx-4xxx-yxxx-xxxxxxxxxxxx`, where `x` is a random lowercase hexadecimal digit and `y` is one of `8`, `9`, `a`, or `b`.

115.6 Remarks

- The GUID is generated using a cryptographically secure random number generator.
- Each call produces a statistically unique value. Collisions are possible but astronomically unlikely.
- Method names are case-insensitive.

115.7 Example

```
<%
Option Explicit
Dim ax, guid
Set ax = Server.CreateObject("G3AXON.FUNCTIONS")

guid = ax.AxGenerateGuid()
Response.Write guid
' Output example: f47ac10b-58cc-4372-a567-0e02b2c3d479

Set ax = Nothing
%>
```

115.8 API Reference

- **Object**: `G3AXON.FUNCTIONS`
- **Method**: `AxGenerateGuid`
- **Arguments**: None
- **Returns**: String — 36-character RFC 4122 v4 GUID

116 Get the Default CSS Stylesheet Content

116.1 Overview

Returns the text content of the CSS file configured in the `axfunctions.ax_default_css_path` setting of `config/axonasp.toml`.

116.2 Prerequisites

Instantiate the library with `Server.CreateObject("G3AXON.FUNCTIONS")`.

Set the CSS file path in `config/axonasp.toml`:

```
[axfunctions]
ax_default_css_path = "./www/css/axonasp.css"
```

116.3 Syntax

```
result = ax.AxGetDefaultCss()
```

116.4 Parameters

This method does not accept any parameters.

116.5 Return Value

- **String**: The raw CSS text from the configured file.
- **String (empty)**: Returned when `ax_default_css_path` is not configured or the file cannot be read.

116.6 Remarks

- The file path is resolved relative to the AxonASP executable directory.
- This method allows ASP pages to inline the system stylesheet without hardcoding file paths.
- Method names are case-insensitive.

116.7 Example

```
<%
Option Explicit
Dim ax, css
Set ax = Server.CreateObject("G3AXON.FUNCTIONS")

css = ax.AxGetDefaultCss()

If css <> "" Then
    Response.Write "<style>" & vbCrLf & css & vbCrLf & "</style>"
End If

Set ax = Nothing
%>
```

116.8 API Reference

- **Object**: `G3AXON.FUNCTIONS`
- **Method**: `AxGetDefaultCss`
- **Arguments**: None
- **Returns**: String — CSS file content, or empty string if not configured or unreadable

117 Use the G3CRYPTO Library

117.1 Overview

G3CRYPTO provides native cryptographic operations for G3Pix AxonASP applications. It supports message digests, HMAC signatures, password hashing with bcrypt, PBKDF2 key derivation, secure random generation, and UUID creation.

117.2 Prerequisites

No external dependencies are required.

117.3 Syntax

```
Set crypto = Server.CreateObject("G3CRYPTO")
```

117.4 How it Works

The object computes digests either as hex strings (for methods such as SHA256) or as raw byte arrays (for ComputeHash).

The object stores state for:

- Hash: Last raw digest bytes produced by digest operations that update internal hash state.
- BCryptCost: Configurable work factor used by bcrypt password hashing.

Password operations use bcrypt. Key derivation uses PBKDF2-HMAC-SHA256. Random generators use cryptographically secure OS randomness.

117.5 API Reference

117.5.1 Object

- **ProgID**: G3CRYPTO

117.5.2 Method Categories

- **Digest Methods**: MD5, SHA1, SHA256, SHA384, SHA512, SHA3_256, SHA3_512, Blake2b256, Blake2b512
- **Digest Bytes Method**: ComputeHash
- **Password Methods**: HashPassword, VerifyPassword, SetBCryptCost, GetBCryptCost
- **MAC/KDF Methods**: HmacSha256, HmacSha512, Pbkdf2Sha256
- **Random/Identifier Methods**: RandomBytes, RandomHex, RandomBase64, UUID
- **State Method**: Initialize

117.5.3 Properties

- **BCryptCost** (Read/Write): bcrypt work factor.
- **CanReuseTransform** (Read-only): Always True.
- **Hash** (Read-only): Last raw digest bytes.
- **HashSize** (Read-only): Digest size in bits for the current context.

117.6 Code Example

The following example shows password hashing, verification, and secure token generation.

```
<%
Option Explicit
Dim crypto, password, hash, token
Set crypto = Server.CreateObject("G3CRYPTO")

password = "UserSecret123!"
hash = crypto.HashPassword(password)
```

```
Response.Write "Password Hash: " & hash & "<br>"

If crypto.VerifyPassword(password, hash) Then
    Response.Write "Verification successful<br>"
End If

token = crypto.RandomHex(32)
Response.Write "Random Token: " & token & "<br>"

Set crypto = Nothing
%>
```

118 G3CRYPTO Methods

118.1 Overview

This page summarizes every method exposed by G3CRYPTO.

118.2 Methods

Method	Returns	Description
`Blake2b256(input)`	String	64-character lowercase hexadecimal BLAKE2b-256 digest.
`Blake2b512(input)`	String	128-character lowercase hexadecimal BLAKE2b-512 digest.
`ComputeHash(input [, algorithm])`	Array	Raw digest bytes as a zero-based VBScript byte array.
`GetBCryptCost()`	Integer	Current bcrypt work factor.
`HashPassword(password)`	String	bcrypt hash string, or empty string on failure.
`HmacSha256(data, key)`	String	64-character lowercase hexadecimal HMAC-SHA256 digest.
`HmacSha512(data, key)`	String	128-character lowercase hexadecimal HMAC-SHA512 digest.
`Initialize()`	Empty	Clears internal last-hash state.
`MD5(input)`	String	32-character lowercase hexadecimal MD5 digest.

Method	Returns	Description
`Pbkdf2Sha256(password, salt [, iterations] [, keyLength])`	String	Lowercase hexadecimal derived key.
`RandomBase64([size])`	String	Base64-encoded cryptographically secure random bytes.
`RandomBytes([size])`	Array	Raw cryptographically secure random bytes as a VBScript byte array.
`RandomHex([size])`	String	Lowercase hexadecimal cryptographically secure random bytes.
`SetBCryptCost(cost)`	Boolean	`True` when cost is in range `4..31`; otherwise `False`.
`SHA1(input)`	String	40-character lowercase hexadecimal SHA-1 digest.
`SHA256(input)`	String	64-character lowercase hexadecimal SHA-256 digest.
`SHA3_256(input)`	String	64-character lowercase hexadecimal SHA3-256 digest.
`SHA3_512(input)`	String	128-character lowercase hexadecimal SHA3-512 digest.
`SHA384(input)`	String	96-character lowercase hexadecimal SHA-384 digest.
`SHA512(input)`	String	128-character lowercase hexadecimal SHA-512 digest.
`UUID()`	String	UUID v4 string, or empty string on random-source failure.
`VerifyPassword(password, hash)`	Boolean	`True` on bcrypt match; otherwise `False`.

118.3 Remarks

- Method names are case-insensitive.
- Digest methods return lowercase hexadecimal text unless documented otherwise.
- `ComputeHash` returns raw bytes and updates `Hash` when the algorithm is valid.

119 G3CRYPTO Properties

119.1 Overview

This page lists the properties exposed by `G3CRYPTO`.

119.2 Properties

Property	Access	Type	Description
`BCryptCost`	Read/Write	Integer	bcrypt work factor used by `HashPassword`.
`CanReuseTransform`	Read-only	Boolean	Always `True`.
`Hash`	Read-only	Array	Raw bytes from the most recent digest operation that updates internal hash state.
`HashSize`	Read-only	Integer	Digest size in bits for the current internal hash context.

119.3 Remarks

- Property reads do not perform cryptographic work.
- Setting `BCryptCost` outside `4..31` is ignored.

120 Compute a BLAKE2b-256 Digest

120.1 Overview

Computes the BLAKE2b-256 digest of the input and returns the result as a lowercase hexadecimal string.

120.2 Prerequisites

Instantiate the library with `Server.CreateObject("G3CRYPTO")`.

120.3 Syntax

```
result = crypto.Blake2b256(input)
```

120.4 Parameters

Parameter	Type	Required	Description
input	String or Array	No	Input data as text or a VBScript byte array. When omitted, the method hashes an empty string.

120.5 Return Value

- **String**: 64-character lowercase hexadecimal BLAKE2b-256 digest.

120.6 Remarks

- BLAKE2b is optimized for performance and provides strong cryptographic properties.
- Method names are case-insensitive.

120.7 Example

```
<%
Option Explicit
Dim crypto, digest
Set crypto = Server.CreateObject("G3CRYPTO")

digest = crypto.Blake2b256("Hello AxonASP")
Response.Write digest

Set crypto = Nothing
%>
```

120.8 API Reference

- **Object**: `G3CRYPTO`
- **Method**: `Blake2b256`
- **Arguments**: `input` (String or Array, optional)
- **Returns**: String — 64-character lowercase hexadecimal digest

121 Compute a BLAKE2b-512 Digest

121.1 Overview

Computes the BLAKE2b-512 digest of the input and returns the result as a lowercase hexadecimal string.

121.2 Prerequisites

Instantiate the library with `Server.CreateObject("G3CRYPTO")`.

121.3 Syntax

```
result = crypto.Blake2b512(input)
```

121.4 Parameters

Parameter	Type	Required	Description
input	String or Array	No	Input data as text or a VBScript byte array. When omitted, the method hashes an empty string.

121.5 Return Value

- **String**: 128-character lowercase hexadecimal BLAKE2b-512 digest.

121.6 Remarks

- BLAKE2b-512 is suitable for high-integrity scenarios requiring a 512-bit digest.
- Method names are case-insensitive.

121.7 Example

```
<%
Option Explicit
Dim crypto, digest
Set crypto = Server.CreateObject("G3CRYPTO")

digest = crypto.Blake2b512("Hello AxonASP")
Response.Write digest

Set crypto = Nothing
%>
```

121.8 API Reference

- **Object**: G3CRYPTO

- **Method**: `Blake2b512`
- **Arguments**: `input` (String or Array, optional)
- **Returns**: String — 128-character lowercase hexadecimal digest

122 Compute a Hash as a Byte Array

122.1 Overview

Computes a hash digest and returns the raw digest bytes as a VBScript array.

122.2 Prerequisites

Instantiate the library with `Server.CreateObject("G3CRYPTO")`.

122.3 Syntax

```
result = crypto.ComputeHash(input [, algorithm])
```

122.4 Parameters

Parameter	Type	Required	Description
input	String or Array	No	Input data as text or a VBScript byte array. When omitted, an empty string is hashed.
algorithm	String	No	Hash algorithm name. If omitted on a standard `G3CRYPTO` instance, `sha256` is used.

Supported algorithm names: `md5`, `sha1`, `sha256`, `sha384`, `sha512`, `sha3_256`, `sha3_512`, `blake2b256`, `blake2b512`.

122.5 Return Value

- **Array**: Zero-based VBScript byte array containing the digest.
- **Array (empty)**: Returned when the algorithm name is not recognized.

122.6 Remarks

- This method updates the `Hash` property with the same bytes when a valid algorithm is used.
- When the algorithm is invalid, `Hash` is not updated.
- Method names are case-insensitive.

122.7 Example

```
<%
Option Explicit
Dim crypto, hashBytes
Set crypto = Server.CreateObject("G3CRYPTO")

hashBytes = crypto.ComputeHash("Hello World", "sha256")
Response.Write "Digest bytes: " & (UBound(hashBytes) + 1)

Set crypto = Nothing
%>
```

122.8 API Reference

- **Object**: `G3CRYPTO`
- **Method**: `ComputeHash`
- **Arguments**: `input` (String or Array, optional), `algorithm` (String, optional)
- **Returns**: Array — digest bytes, or empty array for unsupported algorithm

123 Get the bcrypt Cost

123.1 Overview

Returns the current bcrypt work factor.

123.2 Prerequisites

Instantiate the library with `Server.CreateObject("G3CRYPTO")`.

123.3 Syntax

```
result = crypto.GetBCryptCost()
```

123.4 Parameters

This method does not accept any parameters.

123.5 Return Value

- **Integer**: Current bcrypt cost value.

123.6 Remarks

- Default value is `10`.
- This value is the same value exposed by the `BCryptCost` property.
- Method names are case-insensitive.

123.7 Example

```
<%
Option Explicit
Dim crypto, cost
Set crypto = Server.CreateObject("G3CRYPTO")

cost = crypto.GetBCryptCost()
Response.Write "Current bcrypt cost: " & cost

Set crypto = Nothing
%>
```

123.8 API Reference

- **Object**: `G3CRYPTO`
- **Method**: `GetBCryptCost`
- **Arguments**: None
- **Returns**: Integer — current bcrypt work factor

124 Hash a Password with bcrypt

124.1 Overview

Generates a bcrypt hash string for a plain-text password using the current bcrypt cost configuration.

124.2 Prerequisites

Instantiate the library with `Server.CreateObject("G3CRYPTO")`.

124.3 Syntax

```
result = crypto.HashPassword(password)
```

124.4 Parameters

Parameter	Type	Required	Description
password	String	No	Plain-text password to hash. When omitted, an empty string is hashed.

124.5 Return Value

- **String**: bcrypt hash string in the `$2a$<cost>$...` format.
- **String (empty)**: Returned when bcrypt hashing fails.

124.6 Remarks

- The current `BCryptCost` value controls the bcrypt work factor.
- Use `VerifyPassword` to validate user input against a stored bcrypt hash.
- Method names are case-insensitive.

124.7 Example

```
<%
Option Explicit
Dim crypto, hashed
Set crypto = Server.CreateObject("G3CRYPTO")

hashed = crypto.HashPassword("mySecretPassword")
Response.Write hashed

Set crypto = Nothing
%>
```

124.8 API Reference

- **Object**: `G3CRYPTO`
- **Method**: `HashPassword`
- **Arguments**: `password` (String, optional)
- **Returns**: String — bcrypt hash string, or empty string on failure

125 Compute an HMAC-SHA256 Signature

125.1 Overview

Computes an HMAC signature using SHA-256 and returns the digest as a lowercase hexadecimal string.

125.2 Prerequisites

Instantiate the library with `Server.CreateObject("G3CRYPTO")`.

125.3 Syntax

```
result = crypto.HmacSha256(data, key)
```

125.4 Parameters

Parameter	Type	Required	Description
data	String	Yes	Message payload to sign.
key	String	Yes	Secret key for HMAC generation.

125.5 Return Value

- **String**: 64-character lowercase hexadecimal HMAC-SHA256 digest.
- **String (empty)**: Returned when required arguments are missing.

125.6 Remarks

- On success, this method also updates the `Hash` property with raw digest bytes.
- Method names are case-insensitive.

125.7 Example

```
<%
Option Explicit
Dim crypto, signature
Set crypto = Server.CreateObject("G3CRYPTO")

signature = crypto.HmacSha256("Hello World", "mySecretKey")
Response.Write signature

Set crypto = Nothing
%>
```

125.8 API Reference

- **Object**: `G3CRYPTO`
- **Method**: `HmacSha256`
- **Arguments**: `data` (String, required), `key` (String, required)
- **Returns**: String — 64-character lowercase hexadecimal digest, or empty string when arguments are missing

126 Compute an HMAC-SHA512 Signature

126.1 Overview

Computes an HMAC signature using SHA-512 and returns the digest as a lowercase hexadecimal string.

126.2 Prerequisites

Instantiate the library with `Server.CreateObject("G3CRYPTO")`.

126.3 Syntax

```
result = crypto.HmacSha512(data, key)
```

126.4 Parameters

Parameter	Type	Required	Description

Parameter	Type	Required	Description
data	String	Yes	Message payload to sign.
key	String	Yes	Secret key for HMAC generation.

126.5 Return Value

- **String**: 128-character lowercase hexadecimal HMAC-SHA512 digest.
- **String (empty)**: Returned when required arguments are missing.

126.6 Remarks

- On success, this method also updates the `Hash` property with raw digest bytes.
- Method names are case-insensitive.

126.7 Example

```
<%
Option Explicit
Dim crypto, signature
Set crypto = Server.CreateObject("G3CRYPTO")

signature = crypto.HmacSha512("Hello World", "mySecretKey")
Response.Write signature

Set crypto = Nothing
%>
```

126.8 API Reference

- **Object**: `G3CRYPTO`
- **Method**: `HmacSha512`
- **Arguments**: `data` (String, required), `key` (String, required)
- **Returns**: String — 128-character lowercase hexadecimal digest, or empty string when arguments are missing

127 Reset the Internal Hash State

127.1 Overview

Clears the internal last-hash buffer maintained by the current `G3CRYPTO` object instance.

127.2 Prerequisites

Instantiate the library with `Server.CreateObject("G3CRYPTO")`.

127.3 Syntax

```
result = crypto.Initialize()
```

127.4 Parameters

This method does not accept any parameters.

127.5 Return Value

- **Empty**: Always returns `Empty`.

127.6 Remarks

- This method clears the value returned by the `Hash` property.
- This method does not change `BCryptCost`.
- Method names are case-insensitive.

127.7 Example

```
<%
Option Explicit
Dim crypto, bytes
Set crypto = Server.CreateObject("G3CRYPTO")

bytes = crypto.ComputeHash("Some data", "sha256")
crypto.Initialize()

Set crypto = Nothing
%>
```

127.8 API Reference

- **Object**: `G3CRYPTO`
- **Method**: `Initialize`
- **Arguments**: None
- **Returns**: Empty

128 Compute an MD5 Digest

128.1 Overview

Computes the MD5 digest of the input and returns the result as a lowercase hexadecimal string.

128.2 Prerequisites

Instantiate the library with `Server.CreateObject("G3CRYPTO")`.

128.3 Syntax

```
result = crypto.MD5(input)
```

128.4 Parameters

Parameter	Type	Required	Description
input	String or Array	No	Input data as text or a VBScript byte array. When omitted, the method hashes an empty string.

128.5 Return Value

- **String**: 32-character lowercase hexadecimal MD5 digest.

128.6 Remarks

- MD5 is fast and useful for checksums but is not recommended for password hashing or collision-resistant security scenarios.
- The method accepts byte arrays and string inputs.
- Method names are case-insensitive.

128.7 Example

```
<%
Option Explicit
Dim crypto, digest
Set crypto = Server.CreateObject("G3CRYPTO")

digest = crypto.MD5("Hello World")
Response.Write digest
' Output: b10a8db164e0754105b7a99be72e3fe5

Set crypto = Nothing
%>
```

128.8 API Reference

- **Object**: `G3CRYPTO`
- **Method**: `MD5`
- **Arguments**: `input` (String or Array, optional)
- **Returns**: String — 32-character lowercase hexadecimal digest

129 Derive a Key with PBKDF2-SHA256

129.1 Overview

Derives a key from a password and salt using PBKDF2 with HMAC-SHA256.

129.2 Prerequisites

Instantiate the library with `Server.CreateObject("G3CRYPTO")`.

129.3 Syntax

```
result = crypto.Pbkdf2Sha256(password, salt [, iterations] [, keyLength])
```

129.4 Parameters

Parameter	Type	Required	Description
password	String	Yes	Password or passphrase input.
salt	String	Yes	Salt value. Use a unique random salt per password.
iterations	Integer	No	Iteration count. Values less than or equal to `0` are replaced with `100000`.
keyLength	Integer	No	Derived key length in bytes. Values less than or equal to `0` are replaced with `32`.

129.5 Return Value

- **String**: Lowercase hexadecimal representation of the derived key. Length is `keyLength * 2` characters after normalization.
- **String (empty)**: Returned when required arguments are missing.

129.6 Remarks

- On success, this method also updates the `Hash` property with raw derived-key bytes.
- PBKDF2 is intentionally CPU-intensive and suitable for password-based key derivation.
- Method names are case-insensitive.

129.7 Example

```
<%
Option Explicit
Dim crypto, salt, keyHex
```

```
Set crypto = Server.CreateObject("G3CRYPTO")

salt = "user-specific-random-salt"
keyHex = crypto.Pbkdf2Sha256("mySecretPassword", salt, 150000, 32)
Response.Write keyHex

Set crypto = Nothing
%>
```

129.8 API Reference

- **Object**: `G3CRYPTO`
- **Method**: `Pbkdf2Sha256`
- **Arguments**: `password` (String, required), `salt` (String, required), `iterations` (Integer, optional), `keyLength` (Integer, optional)
- **Returns**: String — lowercase hexadecimal derived key, or empty string when required arguments are missing

130 Generate a Secure Random Base64 String

130.1 Overview

Generates cryptographically secure random bytes and returns them as a Base64 string.

130.2 Prerequisites

Instantiate the library with `Server.CreateObject("G3CRYPTO")`.

130.3 Syntax

```
result = crypto.RandomBase64([size])
```

130.4 Parameters

Parameter	Type	Required	Description
size	Integer	No	Number of random bytes to generate. Default is `32`. Negative values are normalized to `0`.

130.5 Return Value

- **String**: Base64-encoded random data.
- **String (empty)**: Returned when `size` is `0` or random generation fails.

130.6 Remarks

- Base64 output length depends on the byte size and padding.
- Method names are case-insensitive.

130.7 Example

```
<%
Option Explicit
Dim crypto, token
Set crypto = Server.CreateObject("G3CRYPTO")

token = crypto.RandomBase64(16)
Response.Write token

Set crypto = Nothing
%>
```

130.8 API Reference

- **Object**: `G3CRYPTO`
- **Method**: `RandomBase64`
- **Arguments**: `size` (Integer, optional)
- **Returns**: String — Base64 output, or empty string on failure/zero length

131 Generate Secure Random Bytes

131.1 Overview

Generates cryptographically secure random data and returns it as a VBScript byte array.

131.2 Prerequisites

Instantiate the library with `Server.CreateObject("G3CRYPTO")`.

131.3 Syntax

```
result = crypto.RandomBytes([size])
```

131.4 Parameters

Parameter	Type	Required	Description
size	Integer	No	Number of bytes to generate. Default is `32`. Negative values are normalized to `0`.

131.5 Return Value

- **Array**: Zero-based VBScript byte array containing random bytes.
- **Array (empty)**: Returned when `size` is `0` or when random generation fails.

131.6 Remarks

- Use this method for salts, nonces, key material, and opaque tokens.
- Method names are case-insensitive.

131.7 Example

```
<%
Option Explicit
Dim crypto, bytes
Set crypto = Server.CreateObject("G3CRYPTO")

bytes = crypto.RandomBytes(16)
If UBound(bytes) >= 0 Then
    Response.Write "First byte: " & bytes(0)
End If

Set crypto = Nothing
%>
```

131.8 API Reference

- **Object**: `G3CRYPTO`
- **Method**: `RandomBytes`
- **Arguments**: `size` (Integer, optional)
- **Returns**: Array — random bytes, or empty array on failure/zero length

132 Generate a Secure Random Hex String

132.1 Overview

Generates cryptographically secure random bytes and returns them as a lowercase hexadecimal string.

132.2 Prerequisites

Instantiate the library with `Server.CreateObject("G3CRYPTO")`.

132.3 Syntax

```
result = crypto.RandomHex([size])
```

132.4 Parameters

Parameter	Type	Required	Description

Parameter	Type	Required	Description
size	Integer	No	Number of random bytes to generate. Default is 32. Negative values are normalized to 0.

132.5 Return Value

- **String**: Lowercase hexadecimal string with `size * 2` characters after normalization.
- **String (empty)**: Returned when `size` is `0` or random generation fails.

132.6 Remarks

- This method is useful for token generation where hex encoding is preferred.
- Method names are case-insensitive.

132.7 Example

```
<%
Option Explicit
Dim crypto, token
Set crypto = Server.CreateObject("G3CRYPTO")

token = crypto.RandomHex(16)
Response.Write token
' 32 hex characters

Set crypto = Nothing
%>
```

132.8 API Reference

- **Object**: `G3CRYPTO`
- **Method**: `RandomHex`
- **Arguments**: `size` (Integer, optional)
- **Returns**: String — lowercase hex output, or empty string on failure/zero length

133 Set the bcrypt Cost

133.1 Overview

Sets the bcrypt work factor used by `HashPassword`.

133.2 Prerequisites

Instantiate the library with `Server.CreateObject("G3CRYPTO")`.

133.3 Syntax

```
result = crypto.SetBCryptCost(cost)
```

133.4 Parameters

Parameter	Type	Required	Description
cost	Integer	Yes	bcrypt work factor. Valid range is 4 to 31.

133.5 Return Value

- **Boolean `True`**: Cost was accepted and applied.
- **Boolean `False`**: Cost is outside `4..31` or argument is missing.

133.6 Remarks

- Default bcrypt cost is `10`.
- Higher cost increases password hashing time and CPU usage.
- Method names are case-insensitive.

133.7 Example

```
<%
Option Explicit
Dim crypto, ok
Set crypto = Server.CreateObject("G3CRYPTO")

ok = crypto.SetBCryptCost(12)
If ok Then
    Response.Write "bcrypt cost updated to 12"
Else
    Response.Write "Invalid bcrypt cost"
End If

Set crypto = Nothing
%>
```

133.8 API Reference

- **Object**: `G3CRYPTO`
- **Method**: `SetBCryptCost`
- **Arguments**: `cost` (Integer, required)
- **Returns**: Boolean — `True` on success, `False` on invalid value

134 Compute a SHA-1 Digest

134.1 Overview

Computes the SHA-1 digest of the input and returns the result as a lowercase hexadecimal string.

134.2 Prerequisites

Instantiate the library with `Server.CreateObject("G3CRYPTO")`.

134.3 Syntax

```
result = crypto.SHA1(input)
```

134.4 Parameters

Parameter	Type	Required	Description
input	String or Array	No	Input data as text or a VBScript byte array. When omitted, the method hashes an empty string.

134.5 Return Value

- **String**: 40-character lowercase hexadecimal SHA-1 digest.

134.6 Remarks

- SHA-1 is retained for compatibility and non-security-sensitive checks.
- Use SHA-256 or stronger algorithms for new security-critical features.
- Method names are case-insensitive.

134.7 Example

```
<%
Option Explicit
Dim crypto, digest
Set crypto = Server.CreateObject("G3CRYPTO")

digest = crypto.SHA1("Hello World")
Response.Write digest
' Output: 0a4d55a8d778e5022fab701977c5d840bbc486d0

Set crypto = Nothing
%>
```

134.8 API Reference

- **Object**: `G3CRYPTO`
- **Method**: `SHA1`
- **Arguments**: `input` (String or Array, optional)
- **Returns**: String — 40-character lowercase hexadecimal digest

135 Compute a SHA-256 Digest

135.1 Overview

Computes the SHA-256 digest of the input and returns the result as a lowercase hexadecimal string.

135.2 Prerequisites

Instantiate the library with `Server.CreateObject("G3CRYPTO")`.

135.3 Syntax

```
result = crypto.SHA256(input)
```

135.4 Parameters

Parameter	Type	Required	Description
input	String or Array	No	Input data as text or a VBScript byte array. When omitted, the method hashes an empty string.

135.5 Return Value

- **String**: 64-character lowercase hexadecimal SHA-256 digest.

135.6 Remarks

- SHA-256 is a recommended default for modern integrity checks.
- The method accepts both textual input and binary arrays.
- Method names are case-insensitive.

135.7 Example

```
<%
Option Explicit
Dim crypto, digest
Set crypto = Server.CreateObject("G3CRYPTO")
```

```
digest = crypto.SHA256("Hello, AxonASP!")
Response.Write digest

Set crypto = Nothing
%>
```

135.8 API Reference

- **Object**: `G3CRYPTO`
- **Method**: `SHA256`
- **Arguments**: `input` (String or Array, optional)
- **Returns**: String — 64-character lowercase hexadecimal digest

136 Compute a SHA3-256 Digest

136.1 Overview

Computes the SHA3-256 digest of the input and returns the result as a lowercase hexadecimal string.

136.2 Prerequisites

Instantiate the library with `Server.CreateObject("G3CRYPTO")`.

136.3 Syntax

```
result = crypto.SHA3_256(input)
```

136.4 Parameters

Parameter	Type	Required	Description
input	String or Array	No	Input data as text or a VBScript byte array. When omitted, the method hashes an empty string.

136.5 Return Value

- **String**: 64-character lowercase hexadecimal SHA3-256 digest.

136.6 Remarks

- SHA-3 is a modern hash family defined in FIPS 202.
- Method names are case-insensitive.

136.7 Example

```
<%
Option Explicit
Dim crypto, digest
Set crypto = Server.CreateObject("G3CRYPTO")

digest = crypto.SHA3_256("AxonASP Secure Data")
Response.Write digest

Set crypto = Nothing
%>
```

136.8 API Reference

- **Object**: `G3CRYPTO`
- **Method**: `SHA3_256`
- **Arguments**: `input` (String or Array, optional)
- **Returns**: String — 64-character lowercase hexadecimal digest

137 Compute a SHA3-512 Digest

137.1 Overview

Computes the SHA3-512 digest of the input and returns the result as a lowercase hexadecimal string.

137.2 Prerequisites

Instantiate the library with `Server.CreateObject("G3CRYPTO")`.

137.3 Syntax

```
result = crypto.SHA3_512(input)
```

137.4 Parameters

Parameter	Type	Required	Description
input	String or Array	No	Input data as text or a VBScript byte array. When omitted, the method hashes an empty string.

137.5 Return Value

- **String**: 128-character lowercase hexadecimal SHA3-512 digest.

137.6 Remarks

- SHA3-512 is a high-strength option in the SHA-3 family.
- Method names are case-insensitive.

137.7 Example

```
<%
Option Explicit
Dim crypto, digest
Set crypto = Server.CreateObject("G3CRYPTO")

digest = crypto.SHA3_512("Ultra Secure Payload")
Response.Write digest

Set crypto = Nothing
%>
```

137.8 API Reference

- **Object**: `G3CRYPTO`
- **Method**: `SHA3_512`
- **Arguments**: `input` (String or Array, optional)
- **Returns**: String — 128-character lowercase hexadecimal digest

138 Compute a SHA-384 Digest

138.1 Overview

Computes the SHA-384 digest of the input and returns the result as a lowercase hexadecimal string.

138.2 Prerequisites

Instantiate the library with `Server.CreateObject("G3CRYPTO")`.

138.3 Syntax

```
result = crypto.SHA384(input)
```

138.4 Parameters

Parameter	Type	Required	Description
input	String or Array	No	Input data as text or a VBScript byte array. When omitted, the method hashes an

Parameter	Type	Required	Description
			empty string.

138.5 Return Value

- **String**: 96-character lowercase hexadecimal SHA-384 digest.

138.6 Remarks

- SHA-384 is part of the SHA-2 family and provides a larger digest than SHA-256.
- Method names are case-insensitive.

138.7 Example

```
<%
Option Explicit
Dim crypto, digest
Set crypto = Server.CreateObject("G3CRYPTO")

digest = crypto.SHA384("Secure Data Transmission")
Response.Write digest

Set crypto = Nothing
%>
```

138.8 API Reference

- **Object**: `G3CRYPTO`
- **Method**: `SHA384`
- **Arguments**: `input` (String or Array, optional)
- **Returns**: String — 96-character lowercase hexadecimal digest

139 Compute a SHA-512 Digest

139.1 Overview

Computes the SHA-512 digest of the input and returns the result as a lowercase hexadecimal string.

139.2 Prerequisites

Instantiate the library with `Server.CreateObject("G3CRYPTO")`.

139.3 Syntax

```
result = crypto.SHA512(input)
```

139.4 Parameters

Parameter	Type	Required	Description

Parameter	Type	Required	Description
input	String or Array	No	Input data as text or a VBScript byte array. When omitted, the method hashes an empty string.

139.5 Return Value

- **String**: 128-character lowercase hexadecimal SHA-512 digest.

139.6 Remarks

- SHA-512 provides a large digest size for strong integrity requirements.
- Method names are case-insensitive.

139.7 Example

```
<%
Option Explicit
Dim crypto, digest
Set crypto = Server.CreateObject("G3CRYPTO")

digest = crypto.SHA512("Maximum Security Hash")
Response.Write digest

Set crypto = Nothing
%>
```

139.8 API Reference

- **Object**: `G3CRYPTO`
- **Method**: `SHA512`
- **Arguments**: `input` (String or Array, optional)
- **Returns**: String — 128-character lowercase hexadecimal digest

140 Generate a UUID v4

140.1 Overview

Generates one RFC 4122 version 4 UUID string.

140.2 Prerequisites

Instantiate the library with `Server.CreateObject("G3CRYPTO")`.

140.3 Syntax

```
result = crypto.UUID()
```

140.4 Parameters

This method does not accept any parameters.

140.5 Return Value

- **String**: UUID string in the `8-4-4-4-12` format, using lowercase hexadecimal characters.
- **String (empty)**: Returned when cryptographic random generation fails.

140.6 Remarks

- Use this method for unique identifiers where randomness and uniqueness are required.
- Method names are case-insensitive.

140.7 Example

```
<%
Option Explicit
Dim crypto, id
Set crypto = Server.CreateObject("G3CRYPTO")

id = crypto.UUID()
Response.Write id

Set crypto = Nothing
%>
```

140.8 API Reference

- **Object**: `G3CRYPTO`
- **Method**: `UUID`
- **Arguments**: None
- **Returns**: String — UUID v4 text, or empty string on random-source failure

141 Verify a Password Against a bcrypt Hash

141.1 Overview

Checks whether a plain-text password matches a stored bcrypt hash.

141.2 Prerequisites

Instantiate the library with `Server.CreateObject("G3CRYPTO")`.

141.3 Syntax

```
result = crypto.VerifyPassword(password, hash)
```

141.4 Parameters

Parameter	Type	Required	Description
password	String	Yes	User-supplied plain-text password.
hash	String	Yes	Stored bcrypt hash string.

141.5 Return Value

- **Boolean `True`**: Password matches the bcrypt hash.
- **Boolean `False`**: Password does not match, hash is invalid, or required arguments are missing.

141.6 Remarks

- This method is the recommended password verification path for authentication flows.
- bcrypt hash parsing and verification are handled internally.
- Method names are case-insensitive.

141.7 Example

```
<%
Option Explicit
Dim crypto, storedHash, userPass
Set crypto = Server.CreateObject("G3CRYPTO")

storedHash = "$2a$10$WmEynFf7ivBvJ9fD2X4aj.7YyUw8lnjxudlEE4u7YwY3QJ7V6dNl
m"
userPass = Request.Form("password")

If crypto.VerifyPassword(userPass, storedHash) Then
    Response.Write "Login Success"
Else
    Response.Write "Invalid Credentials"
End If

Set crypto = Nothing
%>
```

141.8 API Reference

- **Object**: `G3CRYPTO`
- **Method**: `VerifyPassword`
- **Arguments**: `password` (String, required), `hash` (String, required)

- **Returns**: Boolean — `True` when password matches; `False` otherwise

142 BCryptCost Property

142.1 Overview

Gets or sets the bcrypt work factor used by `HashPassword`.

142.2 Prerequisites

Instantiate the library with `Server.CreateObject("G3CRYPTO")`.

142.3 Syntax

```
' Get current cost
cost = crypto.BCryptCost

' Set new cost
crypto.BCryptCost = 12
```

142.4 Return Value

- **Integer**: Current bcrypt work factor.

142.5 Remarks

- Default value is `10`.
- Valid range is `4..31`.
- Setting values outside `4..31` has no effect.
- Higher values increase CPU time for hashing and verification.

142.6 API Reference

- **Object**: `G3CRYPTO`
- **Property**: `BCryptCost`
- **Access**: Read/Write
- **Type**: Integer
- **Valid Set Range**: `4..31`

142.7 Code Example

```
<%
Option Explicit
Dim crypto
Set crypto = Server.CreateObject("G3CRYPTO")

crypto.BCryptCost = 12
Response.Write "New BCrypt Cost: " & crypto.BCryptCost
```

```
Set crypto = Nothing
%>
```

143 CanReuseTransform Property

143.1 Overview

Indicates whether the crypto transform can be reused across operations.

143.2 Prerequisites

Instantiate the library with `Server.CreateObject("G3CRYPTO")`.

143.3 Syntax

```
isReusable = crypto.CanReuseTransform
```

143.4 Return Value

- **Boolean**: Always `True`.

143.5 Remarks

- This property is provided for compatibility.
- It is not a runtime capability probe because the value is constant.

143.6 API Reference

- **Object**: `G3CRYPTO`
- **Property**: `CanReuseTransform`
- **Access**: Read-only
- **Type**: Boolean (`True`)

143.7 Code Example

```
<%
Option Explicit
Dim crypto
Set crypto = Server.CreateObject("G3CRYPTO")

If crypto.CanReuseTransform Then
    Response.Write "Transform is reusable"
End If

Set crypto = Nothing
%>
```

144 Hash Property

144.1 Overview

Returns the raw byte array from the latest digest operation that updates internal hash state.

144.2 Prerequisites

Instantiate the library with `Server.CreateObject("G3CRYPTO")`.

144.3 Syntax

```
byteArray = crypto.Hash
```

144.4 Return Value

- **Array**: Zero-based VBScript byte array.
- **Array (empty)**: Returned when no digest bytes are currently stored.

144.5 Remarks

- Updated by digest operations that store internal bytes, including `ComputeHash`, `HmacSha256`, `HmacSha512`, and `Pbkdf2Sha256`.
- Cleared by `Initialize`.
- Hex-returning digest methods such as `SHA256` return text and do not update this property.

144.6 API Reference

- **Object**: `G3CRYPTO`
- **Property**: `Hash`
- **Access**: Read-only
- **Type**: Array (byte values)

144.7 Code Example

```
<%
Option Explicit
Dim crypto, rawHash, i
Set crypto = Server.CreateObject("G3CRYPTO")

crypto.ComputeHash "Binary Output Test", "sha256"
rawHash = crypto.Hash

Response.Write "First 4 bytes: "
For i = 0 To 3
    Response.Write Hex(rawHash(i)) & " "
Next
```

```
Set crypto = Nothing
%>
```

145 HashSize Property

145.1 Overview

Returns the digest size in bits for the current internal hash context.

145.2 Prerequisites

Instantiate the library with `Server.CreateObject("G3CRYPTO")`.

145.3 Syntax

```
sizeBits = crypto.HashSize
```

145.4 Return Value

- **Integer**: Number of bits in the active digest context.
- **Integer `0`**: Returned when no digest context is available.

145.5 Remarks

- When a default algorithm is configured for the object, this property reflects that algorithm's digest size.
- Otherwise, when raw digest bytes are stored, this property returns `Len(Hash) * 8`.
- Returns `0` when neither algorithm context nor stored hash bytes exist.

145.6 API Reference

- **Object**: `G3CRYPTO`
- **Property**: `HashSize`
- **Access**: Read-only
- **Type**: Integer

145.7 Code Example

```
<%
Option Explicit
Dim crypto
Set crypto = Server.CreateObject("G3CRYPTO")

crypto.ComputeHash "Bit Size Test", "sha256"
Response.Write "Hash Size: " & crypto.HashSize & " bits"
```

```
Set crypto = Nothing
%>
```

146 Use the G3JSON Library

146.1 Overview

Use **G3JSON** to parse JSON text into native G3Pix AxonASP values and serialize native values back to JSON text. The library is built into the runtime and supports object, array, and primitive JSON payloads.

146.2 Prerequisites

- Use a running G3Pix AxonASP environment.
- Create the object with the primary ProgID:

```
Set json = Server.CreateObject("G3JSON")
```

146.3 How It Works

- **Parse** and **LoadFile** convert JSON into AxonASP values.
- JSON objects become **Scripting.Dictionary** instances.
- JSON arrays become VBScript arrays.
- JSON primitives become scalar values (String, Integer/Double, Boolean, or Null).
- **Stringify** converts native values to a JSON string.

146.4 API Reference

146.4.1 Methods

- **LoadFile(path)**: Returns parsed JSON from a file, or **Empty** on read/parse failure.
- **NewArray()**: Returns an empty VBScript array.
- **NewObject()**: Returns a new **Scripting.Dictionary** object.
- **Parse(jsonText)**: Returns parsed JSON value, or **Empty** when input is missing, empty, or invalid.
- **Stringify(value)**: Returns a JSON string, or an empty string when no argument is provided or serialization fails.

146.4.2 Properties

G3JSON does not expose public properties.

146.5 Example

```
<%
Dim json, data, payload
Set json = Server.CreateObject("G3JSON")
```

```
Set data = json.Parse("{""name"": ""G3Pix AxonASP"", ""version"": 2}")
If IsObject(data) Then
    data("version") = data("version") + 1
    payload = json.Stringify(data)
    Response.Write payload
End If

Set data = Nothing
Set json = Nothing
%>
```

147 G3JSON Methods

147.1 Overview

This page summarizes every method exposed by `G3JSON` in G3Pix AxonASP.

147.2 Methods

Method	Returns	Description
`LoadFile(path)`	Dictionary, Array, scalar, Null, or Empty	Reads JSON from a file path and parses it to native values; returns Empty when path is missing, read fails, or JSON is invalid.
`NewArray()`	Array	Creates an empty VBScript array for JSON composition.
`NewObject()`	Scripting.Dictionary	Creates an empty dictionary object for JSON composition.
`Parse(jsonText)`	Dictionary, Array, scalar, Null, or Empty	Parses JSON text to native values; returns Empty when input is missing, empty, or invalid.
`Stringify(value)`	String	Serializes a native value to JSON text; returns empty string when argument is missing or serialization fails.

147.3 Remarks

- Instantiate the library with `Server.CreateObject("G3JSON")`.
- Method names are case-insensitive.
- Parsing methods return Empty on operational failure.

148 G3JSON Properties

148.1 Overview

This page lists properties exposed by `G3JSON`.

148.2 Properties

Property	Access	Type	Description
None	N/A	N/A	`G3JSON` does not expose public properties. Use methods for all operations.

148.3 Remarks

- Instantiate the library with `Server.CreateObject("G3JSON")`.
- Use `methods.md` for callable members.

149 Load JSON from File

149.1 Overview

Reads a JSON file and parses it into a native G3Pix AxonASP value.

149.2 Prerequisites

Instantiate the library with `Server.CreateObject("G3JSON")`.

149.3 Syntax

```
result = json.LoadFile(path)
```

149.4 Parameters

Parameter	Type	Required	Description
path	String	Yes	Virtual or physical path to a JSON file.

149.5 Return Value

- **Scripting.Dictionary**: Returned when the JSON root is an object.

- **Array**: Returned when the JSON root is an array.
- **String** / **Integer** / **Double** / **Boolean** / **Null**: Returned when the JSON root is a primitive.
- **Empty**: Returned when `path` is missing, file read fails, or JSON parsing fails.

149.6 Remarks

- Virtual paths are mapped through the server host when available.
- Parsing behavior matches `Parse`.

149.7 Example

```
<%
Option Explicit
Dim json, cfg
Set json = Server.CreateObject("G3JSON")

Set cfg = json.LoadFile("/config/settings.json")
If IsObject(cfg) Then
    Response.Write cfg("appName")
End If

Set cfg = Nothing
Set json = Nothing
%>
```

149.8 API Reference

- **Object**: `G3JSON`
- **Method**: `LoadFile`
- **Arguments**: `path` (String, required)
- **Returns**: Dictionary, Array, scalar primitive, Null, or Empty on failure

150 Create a New JSON Array Container

150.1 Overview

Creates an empty VBScript array for JSON composition.

150.2 Prerequisites

Instantiate the library with `Server.CreateObject("G3JSON")`.

150.3 Syntax

```
arr = json.NewArray()
```

150.4 Parameters

None.

150.5 Return Value

- **Array**: Empty VBScript array.

150.6 Remarks

- Use this method to build array payloads before calling `Stringify`.

150.7 Example

```
<%
Option Explicit
Dim json, arr
Set json = Server.CreateObject("G3JSON")

arr = json.NewArray()
Response.Write IsArray(arr)

Set json = Nothing
%>
```

150.8 API Reference

- **Object**: `G3JSON`
- **Method**: `NewArray`
- **Arguments**: none
- **Returns**: Array — empty VBScript array

151 Create a New JSON Object Container

151.1 Overview

Creates an empty Scripting.Dictionary object for JSON composition.

151.2 Prerequisites

Instantiate the library with `Server.CreateObject("G3JSON")`.

151.3 Syntax

```
Set obj = json.NewObject()
```

151.4 Parameters

None.

151.5 Return Value

- **Scripting.Dictionary**: Empty dictionary object.

151.6 Remarks

- Use this method to build object payloads before calling `Stringify`.

151.7 Example

```
<%
Option Explicit
Dim json, obj
Set json = Server.CreateObject("G3JSON")

Set obj = json.NewObject()
obj.Add "status", "ok"
Response.Write obj("status")

Set obj = Nothing
Set json = Nothing
%>
```

151.8 API Reference

- **Object**: `G3JSON`
- **Method**: `NewObject`
- **Arguments**: none
- **Returns**: Scripting.Dictionary — empty dictionary object

152 Parse JSON Text

152.1 Overview

Parses JSON text and returns the corresponding native G3Pix AxonASP value.

152.2 Prerequisites

Instantiate the library with `Server.CreateObject("G3JSON")`.

152.3 Syntax

```
result = json.Parse(jsonText)
```

152.4 Parameters

Parameter	Type	Required	Description
jsonText	String	Yes	JSON payload to parse.

152.5 Return Value

- **Scripting.Dictionary**: Returned when the JSON root is an object.
- **Array**: Returned when the JSON root is an array.
- **String** / **Integer** / **Double** / **Boolean** / **Null**: Returned when the JSON root is a primitive.
- **Empty**: Returned when the argument is missing, empty, or invalid JSON.

152.6 Remarks

- Method names are case-insensitive.
- Object and array values are converted recursively.

152.7 Example

```
<%
Option Explicit
Dim json, data
Set json = Server.CreateObject("G3JSON")

Set data = json.Parse("{""name"": ""AxonASP"", ""enabled"": true}")
If IsObject(data) Then
    Response.Write data("name")
End If

Set data = Nothing
Set json = Nothing
%>
```

152.8 API Reference

- **Object**: `G3JSON`
- **Method**: `Parse`
- **Arguments**: `jsonText` (String, required)
- **Returns**: Dictionary, Array, scalar primitive, Null, or Empty on failure

153 Serialize Data to JSON Text

153.1 Overview

Serializes a native G3Pix AxonASP value into JSON text.

153.2 Prerequisites

Instantiate the library with `Server.CreateObject("G3JSON")`.

153.3 Syntax

```
jsonText = json.Stringify(value)
```

153.4 Parameters

Parameter	Type	Required	Description
value	Any	Yes	Dictionary, array, scalar value, Null, or Empty.

153.5 Return Value

- **String**: Serialized JSON text when serialization succeeds.
- **String**: Empty string when the argument is missing.
- **String**: Empty string when serialization fails.

153.6 Remarks

- Dictionary keys become JSON object properties.
- Array values preserve source order.

153.7 Example

```
<%
Option Explicit
Dim json, obj, payload
Set json = Server.CreateObject("G3JSON")

Set obj = json.NewObject()
obj.Add "name", "G3Pix AxonASP"
obj.Add "ok", True

payload = json.Stringify(obj)
Response.Write payload

Set obj = Nothing
Set json = Nothing
%>
```

153.8 API Reference

- **Object**: `G3JSON`
- **Method**: `Stringify`
- **Arguments**: `value` (Any, required)
- **Returns**: String — serialized JSON, or empty string on failure

154 Use the G3DB Library

154.1 Overview

Use **G3DB** to access SQL databases through the native G3Pix AxonASP runtime. The library manages a connection pool, supports parameterized SQL, and exposes result, statement, row, and transaction objects for structured workflows.

154.2 Prerequisites

- Reachable database server or local SQLite file path.
- Valid driver and connection string, or configured environment-based settings.
- Create the object with the primary ProgID:

```
Set db = Server.CreateObject("G3DB")
```

154.3 How It Works

- `Open` and `OpenFromEnv` initialize and validate a pooled connection.
- `Query`, `QueryRow`, `Exec`, and `Prepare` execute SQL against the current connection.
- `Begin` and `BeginTx` create transaction handles.
- Placeholder rewriting converts `?` markers to the active driver format when needed.
- Errors are stored in `LastError` and returned by `GetError`.

154.4 API Reference

154.4.1 Methods

- **Begin()**: Returns a `G3DBTransaction` object on success, otherwise **Empty**.
- **BeginTx([timeoutSeconds, readOnly])**: Returns a `G3DBTransaction` object on success, otherwise **Empty**.
- **Close()**: Returns **True** when close succeeds or when connection is already closed; returns **False** on close error.
- **Exec(sql[, params...])**: Returns a `G3DBResult` object on success, otherwise **Empty**.
- **GetError()**: Returns the current error string.
- **Open(driver, dsn)**: Returns **True** on successful open; **False** otherwise.
- **OpenFromEnv([driver])**: Returns **True** on successful open from config/env; **False** otherwise.
- **Prepare(sql)**: Returns a `G3DBStatement` object on success, otherwise **Empty**.
- **Query(sql[, params...])**: Returns a `G3DBResultSet` object on success, otherwise **Empty**.

- **QueryRow(sql[, params...])**: Returns a `G3DBRow` object on success, otherwise **Empty**.
- **SetConnMaxIdleTime(seconds)**: Returns **Empty**.
- **SetConnMaxLifetime(seconds)**: Returns **Empty**.
- **SetMaxIdleConns(count)**: Returns **Empty**.
- **SetMaxOpenConns(count)**: Returns **Empty**.
- **Stats()**: Returns a `Scripting.Dictionary` when connection is open; otherwise **Empty**.

154.4.2 Properties

- **Driver** (String): Read/write.
- **DSN** (String): Read/write.
- **IsOpen** (Boolean): Read-only.
- **LastError** (String): Read-only.

154.5 Example

```
<%
Dim db, rs
Set db = Server.CreateObject("G3DB")

If db.Open("mysql", "user:pass@tcp(127.0.0.1:3306)/appdb") Then
    Set rs = db.Query("SELECT id, username FROM users WHERE active = ?",
1)
    If Not IsEmpty(rs) Then
        Do While Not rs.EOF
            Response.Write rs("id") & " - " & rs("username") & "<br>"
            rs.MoveNext
        Loop
        rs.Close
    End If
    db.Close
Else
    Response.Write db.LastError
End If

Set db = Nothing
%>
```

155 G3DB Methods

155.1 Overview

This page summarizes every method exposed by `G3DB` in G3Pix AxonASP.

155.2 Methods

Method	Returns	Description
`Begin()`	G3DBTransaction or Empty	Starts a transaction with default options. Returns Empty when connection is not open or transaction start fails.
`BeginTx([timeoutSeconds, readOnly])`	G3DBTransaction or Empty	Starts a transaction with optional timeout/read-only flags. Returns Empty on failure.
`Close()`	Boolean	Returns True on successful close or when already closed; False when close returns an error.
`Exec(sql [, params...])`	G3DBResult or Empty	Executes a non-query statement. Returns Empty when connection is not open, SQL is missing, or execution fails.
`GetError()`	String	Returns the current error message, or empty string when no error is stored.
`Open(driver, dsn)`	Boolean	Returns True when open and ping validation succeed; otherwise False.
`OpenFromEnv([driver])`	Boolean	Opens using configuration/env values. Returns True on success; otherwise False.
`Prepare(sql)`	G3DBStatement or Empty	Creates a prepared statement. Returns Empty on connection/argument/prepare failure.
`Query(sql [, params...])`	G3DBResultSet or Empty	Executes a query and returns a forward-only result set. Returns Empty on failure.

Method	Returns	Description
`QueryRow(sql [, params...])`	G3DBRow or Empty	Executes a query and returns a single-row object for scan methods. Returns Empty on connection or argument failure.
`SetConnMaxIdleTime(seconds)`	Empty	Sets pool idle timeout when connection is open.
`SetConnMaxLifetime(seconds)`	Empty	Sets maximum pool connection lifetime when connection is open.
`SetMaxIdleConns(count)`	Empty	Sets maximum idle pooled connections when connection is open.
`SetMaxOpenConns(count)`	Empty	Sets maximum open pooled connections when connection is open.
`Stats()`	Scripting.Dictionary or Empty	Returns current pool statistics dictionary when connection is open; otherwise Empty.

155.3 Remarks

- Instantiate the library with `Server.CreateObject("G3DB")`.
- Method names are case-insensitive.
- Error details are available through `LastError` and `GetError()`.

156 G3DB Properties

156.1 Overview

This page lists the properties exposed by `G3DB`.

156.2 Properties

Property	Access	Type	Description
`Driver`	Read/Write	String	Normalized database driver name used by the connection object.
`DSN`	Read/Write	String	Current connection

Property	Access	Type	Description
			string stored in the object.
`IsOpen`	Read-only	Boolean	Indicates whether the database pool is currently open.
`LastError`	Read-only	String	Latest error text recorded by the object.

156.3 Remarks

- Instantiate the library with `Server.CreateObject("G3DB")`.
- `Driver` and `DSN` can be assigned before opening a connection.

157 Begin a Transaction

157.1 Overview

Starts a database transaction using default options.

157.2 Prerequisites

Instantiate the library with `Server.CreateObject("G3DB")`.

157.3 Syntax

```
Set tx = db.Begin()
```

157.4 Parameters

None.

157.5 Return Value

- **G3DBTransaction**: Returned when transaction start succeeds.
- **Empty**: Returned when connection is not open or transaction creation fails.

157.6 Remarks

- Aliases `BeginTrans` and `BeginTransaction` call the same runtime path.

157.7 Example

```
<%
Option Explicit
Dim db, tx
Set db = Server.CreateObject("G3DB")
```

```
If db.Open("mysql", "user:pass@tcp(localhost)/db") Then
    Set tx = db.Begin()
    If Not IsEmpty(tx) Then tx.Rollback
    db.Close
End If

Set db = Nothing
%>
```

157.8 API Reference

- **Object**: `G3DB`
- **Method**: `Begin`
- **Arguments**: none
- **Returns**: G3DBTransaction on success, Empty on failure

158 Begin a Transaction with Options

158.1 Overview

Starts a database transaction with optional timeout and read-only configuration.

158.2 Prerequisites

Instantiate the library with `Server.CreateObject("G3DB")`.

158.3 Syntax

```
Set tx = db.BeginTx([timeoutSeconds, readOnly])
```

158.4 Parameters

Parameter	Type	Required	Description
timeoutSeconds	Integer	No	Timeout in seconds. Use `0` or omit for no timeout.
readOnly	Boolean	No	`True` to request read-only transaction mode.

158.5 Return Value

- **G3DBTransaction**: Returned when transaction start succeeds.
- **Empty**: Returned when connection is not open or transaction creation fails.

158.6 Remarks

- Timeout is only applied when `timeoutSeconds` is greater than zero.

158.7 Example

```
<%
Option Explicit
Dim db, tx
Set db = Server.CreateObject("G3DB")

If db.Open("postgres", "host=localhost user=u dbname=app") Then
    Set tx = db.BeginTx(30, True)
    If Not IsEmpty(tx) Then tx.Rollback
    db.Close
End If

Set db = Nothing
%>
```

158.8 API Reference

- **Object**: `G3DB`
- **Method**: `BeginTx`
- **Arguments**: `timeoutSeconds` (Integer, optional), `readOnly` (Boolean, optional)
- **Returns**: G3DBTransaction on success, Empty on failure

159 Close a Database Connection

159.1 Overview

Closes the current database pool managed by the G3DB object.

159.2 Prerequisites

Instantiate the library with `Server.CreateObject("G3DB")`.

159.3 Syntax

```
ok = db.Close()
```

159.4 Parameters

None.

159.5 Return Value

- **Boolean `True`**: Connection closed successfully, or no connection was open.
- **Boolean `False`**: Close operation returned an error.

159.6 Remarks

- On successful close, `IsOpen` becomes `False`.

159.7 Example

```
<%
Option Explicit
Dim db, ok
Set db = Server.CreateObject("G3DB")

db.Open "sqlite", "data.db"
ok = db.Close()
Response.Write ok

Set db = Nothing
%>
```

159.8 API Reference

- **Object**: `G3DB`
- **Method**: `Close`
- **Arguments**: none
- **Returns**: Boolean — `True` on success, `False` on close error

160 Execute a Non-Query SQL Statement

160.1 Overview

Executes a SQL statement that does not return a row cursor.

160.2 Prerequisites

Instantiate the library with `Server.CreateObject("G3DB")`.

160.3 Syntax

```
Set result = db.Exec(sql[, params...])
```

160.4 Parameters

Parameter	Type	Required	Description
sql	String	Yes	SQL statement to execute.
params	Any	No	Positional values for statement placeholders.

160.5 Return Value

- **G3DBResult**: Returned when execution succeeds.
- **Empty**: Returned when connection is not open, SQL is missing, or execution fails.

160.6 Remarks

- Placeholder rewriting is applied for the active driver when required.

160.7 Example

```
<%
Option Explicit
Dim db, result
Set db = Server.CreateObject("G3DB")

If db.Open("mysql", "user:pass@tcp(localhost)/db") Then
    Set result = db.Exec("UPDATE users SET active = ? WHERE id = ?", 1, 4
2)
    If IsEmpty(result) Then Response.Write db.LastError
    db.Close
End If

Set db = Nothing
%>
```

160.8 API Reference

- **Object**: `G3DB`
- **Method**: `Exec`
- **Arguments**: `sql` (String, required), `params...` (Any, optional)
- **Returns**: G3DBResult on success, Empty on failure

161 Get the Current Error Message

161.1 Overview

Returns the latest error text recorded by the G3DB object.

161.2 Prerequisites

Instantiate the library with `Server.CreateObject("G3DB")`.

161.3 Syntax

```
errText = db.GetError()
```

161.4 Parameters

None.

161.5 Return Value

- **String**: Latest recorded error message.
- **String**: Empty string when no error is currently stored.

161.6 Remarks

- Alias: `GetLastError`.
- Returns the same value exposed by the `LastError` property.

161.7 Example

```
<%
Option Explicit
Dim db
Set db = Server.CreateObject("G3DB")

If Not db.Open("mysql", "bad") Then
    Response.Write db.GetError()
End If

Set db = Nothing
%>
```

161.8 API Reference

- **Object**: `G3DB`
- **Method**: `GetError`
- **Arguments**: none
- **Returns**: String — current error message (or empty string)

162 Open a Database Connection

162.1 Overview

Opens a database connection pool and validates connectivity with an internal ping.

162.2 Prerequisites

Instantiate the library with `Server.CreateObject("G3DB")`.

162.3 Syntax

```
ok = db.Open(driver, dsn)
```

162.4 Parameters

Parameter	Type	Required	Description
driver	String	Yes	Database driver name.
dsn	String	Yes	Driver-specific connection string.

162.5 Return Value

- **Boolean `True`**: Connection opened and ping validation succeeded.
- **Boolean `False`**: Arguments are missing, driver is unsupported, connection is already open, or open/ping failed.

162.6 Remarks

- Driver names are normalized before opening.
- On failure, error details are available in `LastError`.

162.7 Example

```
<%
Option Explicit
Dim db, ok
Set db = Server.CreateObject("G3DB")

ok = db.Open("mysql", "user:pass@tcp(127.0.0.1:3306)/app")
If ok Then
    db.Close
Else
    Response.Write db.LastError
End If

Set db = Nothing
%>
```

162.8 API Reference

- **Object**: `G3DB`
- **Method**: `Open`
- **Arguments**: `driver` (String, required), `dsn` (String, required)
- **Returns**: Boolean — `True` on success, `False` on failure

163 Open a Connection from Configuration

163.1 Overview

Opens a database connection using AxonASP configuration and environment overrides.

163.2 Prerequisites

Instantiate the library with `Server.CreateObject("G3DB")`.

163.3 Syntax

```
ok = db.OpenFromEnv([driver])
```

163.4 Parameters

Parameter	Type	Required	Description
driver	String	No	Driver name. Default is `mysql`.

163.5 Return Value

- **Boolean `True`**: Configuration was resolved and connection opened successfully.
- **Boolean `False`**: Driver is unsupported, configuration is missing, or open failed.

163.6 Remarks

- Uses the same connection path as `Open` after DSN resolution.

163.7 Example

```
<%
Option Explicit
Dim db, ok
Set db = Server.CreateObject("G3DB")

ok = db.OpenFromEnv("postgres")
If ok Then
    db.Close
Else
    Response.Write db.LastError
End If

Set db = Nothing
%>
```

163.8 API Reference

- **Object**: `G3DB`
- **Method**: `OpenFromEnv`
- **Arguments**: `driver` (String, optional)
- **Returns**: Boolean — `True` on success, `False` on failure

164 Prepare a Reusable SQL Statement

164.1 Overview

Creates a prepared SQL statement object bound to the active connection.

164.2 Prerequisites

Instantiate the library with `Server.CreateObject("G3DB")`.

164.3 Syntax

```
Set stmt = db.Prepare(sql)
```

164.4 Parameters

Parameter	Type	Required	Description
sql	String	Yes	SQL statement to prepare.

164.5 Return Value

- **G3DBStatement**: Returned when statement preparation succeeds.
- **Empty**: Returned when connection is not open, SQL is missing, or prepare fails.

164.6 Remarks

- Placeholder rewriting is applied before preparation.

164.7 Example

```
<%
Option Explicit
Dim db, stmt
Set db = Server.CreateObject("G3DB")

If db.Open("sqlite", "data.db") Then
    Set stmt = db.Prepare("SELECT name FROM users WHERE id = ?")
    If Not IsEmpty(stmt) Then stmt.Close
    db.Close
End If

Set db = Nothing
%>
```

164.8 API Reference

- **Object**: `G3DB`
- **Method**: `Prepare`
- **Arguments**: `sql` (String, required)
- **Returns**: G3DBStatement on success, Empty on failure

165 Execute a Query and Return a Result Set

165.1 Overview

Executes a SQL query and returns a forward-only G3DBResultSet object.

165.2 Prerequisites

Instantiate the library with `Server.CreateObject("G3DB")`.

165.3 Syntax

```
Set rs = db.Query(sql[, params...])
```

165.4 Parameters

Parameter	Type	Required	Description
sql	String	Yes	SQL query text.
params	Any	No	Positional values for query placeholders.

165.5 Return Value

- **G3DBResultSet**: Returned when query execution succeeds.
- **Empty**: Returned when connection is not open, SQL is missing, or query execution fails.

165.6 Remarks

- Close the result set when processing is complete.

165.7 Example

```
<%
Option Explicit
Dim db, rs
Set db = Server.CreateObject("G3DB")

If db.Open("mysql", "user:pass@tcp(localhost)/db") Then
    Set rs = db.Query("SELECT id FROM users WHERE active = ?", 1)
    If Not IsEmpty(rs) Then rs.Close
    db.Close
End If

Set db = Nothing
%>
```

165.8 API Reference

- **Object**: `G3DB`
- **Method**: `Query`

- **Arguments**: `sql` (String, required), `params...` (Any, optional)
- **Returns**: G3DBResultSet on success, Empty on failure

166 Execute a Query and Return One Row

166.1 Overview

Executes a SQL query and returns a G3DBRow object for single-row scanning.

166.2 Prerequisites

Instantiate the library with `Server.CreateObject("G3DB")`.

166.3 Syntax

```
Set row = db.QueryRow(sql[, params...])
```

166.4 Parameters

Parameter	Type	Required	Description
sql	String	Yes	SQL query text.
params	Any	No	Positional values for query placeholders.

166.5 Return Value

- **G3DBRow**: Returned when query dispatch succeeds.
- **Empty**: Returned when connection is not open or SQL is missing.

166.6 Remarks

- Use `Scan` or `ScanMap` on the returned row object.
- If no row is found, scan methods return Empty.

166.7 Example

```
<%
Option Explicit
Dim db, row, value
Set db = Server.CreateObject("G3DB")

If db.Open("postgres", "host=localhost user=u dbname=app") Then
    Set row = db.QueryRow("SELECT name FROM users WHERE id = ?", 1)
    If Not IsEmpty(row) Then
        value = row.Scan()
        If Not IsEmpty(value) Then Response.Write value
    End If
    db.Close
End If
```

```
Set db = Nothing
%>
```

166.8 API Reference

- **Object**: `G3DB`
- **Method**: `QueryRow`
- **Arguments**: `sql` (String, required), `params...` (Any, optional)
- **Returns**: G3DBRow on dispatch success, Empty on connection/argument failure

167 Set Maximum Connection Idle Time

167.1 Overview

Sets the idle timeout for pooled connections on the active database handle.

167.2 Prerequisites

Instantiate the library with `Server.CreateObject("G3DB")`.

167.3 Syntax

```
db.SetConnMaxIdleTime seconds
```

167.4 Parameters

Parameter	Type	Required	Description
seconds	Integer	Yes	Maximum idle time in seconds.

167.5 Return Value

- **Empty**: Always returned.

167.6 Remarks

- Setting is applied only when a connection is open.

167.7 Example

```
<%
Option Explicit
Dim db
Set db = Server.CreateObject("G3DB")

If db.Open("mysql", "user:pass@tcp(localhost)/db") Then
    db.SetConnMaxIdleTime 120
    db.Close
End If
```

```
Set db = Nothing
%>
```

167.8 API Reference

- **Object**: `G3DB`
- **Method**: `SetConnMaxIdleTime`
- **Arguments**: `seconds` (Integer, required)
- **Returns**: Empty

168 Set Maximum Connection Lifetime

168.1 Overview

Sets the maximum reuse lifetime for pooled connections on the active database handle.

168.2 Prerequisites

Instantiate the library with `Server.CreateObject("G3DB")`.

168.3 Syntax

```
db.SetConnMaxLifetime seconds
```

168.4 Parameters

Parameter	Type	Required	Description
seconds	Integer	Yes	Maximum connection lifetime in seconds.

168.5 Return Value

- **Empty**: Always returned.

168.6 Remarks

- Setting is applied only when a connection is open.

168.7 Example

```
<%
Option Explicit
Dim db
Set db = Server.CreateObject("G3DB")

If db.Open("mysql", "user:pass@tcp(localhost)/db") Then
    db.SetConnMaxLifetime 3600
    db.Close
```

```
End If

Set db = Nothing
%>
```

168.8 API Reference

- **Object**: `G3DB`
- **Method**: `SetConnMaxLifetime`
- **Arguments**: `seconds` (Integer, required)
- **Returns**: Empty

169 Set Maximum Idle Connections

169.1 Overview

Sets the maximum number of idle pooled connections for the active database handle.

169.2 Prerequisites

Instantiate the library with `Server.CreateObject("G3DB")`.

169.3 Syntax

```
db.SetMaxIdleConns count
```

169.4 Parameters

Parameter	Type	Required	Description
count	Integer	Yes	Maximum number of idle pooled connections.

169.5 Return Value

- **Empty**: Always returned.

169.6 Remarks

- Setting is applied only when a connection is open.

169.7 Example

```
<%
Option Explicit
Dim db
Set db = Server.CreateObject("G3DB")

If db.Open("mysql", "user:pass@tcp(localhost)/db") Then
    db.SetMaxIdleConns 10
```

```
    db.Close
End If

Set db = Nothing
%>
```

169.8 API Reference

- **Object**: `G3DB`
- **Method**: `SetMaxIdleConns`
- **Arguments**: `count` (Integer, required)
- **Returns**: Empty

170 Set Maximum Open Connections

170.1 Overview

Sets the maximum number of open pooled connections for the active database handle.

170.2 Prerequisites

Instantiate the library with `Server.CreateObject("G3DB")`.

170.3 Syntax

```
db.SetMaxOpenConns count
```

170.4 Parameters

Parameter	Type	Required	Description
count	Integer	Yes	Maximum number of open pooled connections.

170.5 Return Value

- **Empty**: Always returned.

170.6 Remarks

- Setting is applied only when a connection is open.

170.7 Example

```
<%
Option Explicit
Dim db
Set db = Server.CreateObject("G3DB")
```

```
If db.Open("mysql", "user:pass@tcp(localhost)/db") Then
    db.SetMaxOpenConns 50
    db.Close
End If

Set db = Nothing
%>
```

170.8 API Reference

- **Object**: `G3DB`
- **Method**: `SetMaxOpenConns`
- **Arguments**: `count` (Integer, required)
- **Returns**: Empty

171 Read Database Pool Statistics

171.1 Overview

Returns runtime pool statistics for the active G3DB connection.

171.2 Prerequisites

Instantiate the library with `Server.CreateObject("G3DB")`.

171.3 Syntax

```
Set stats = db.Stats()
```

171.4 Parameters

None.

171.5 Return Value

- **Scripting.Dictionary**: Returned when connection is open.
- **Empty**: Returned when connection is not open.

171.6 Remarks

- Dictionary keys include `MaxOpenConnections`, `OpenConnections`, `InUse`, `Idle`, `WaitCount`, `WaitDurationSeconds`, `MaxIdleClosed`, `MaxIdleTimeClosed`, and `MaxLifetimeClosed`.

171.7 Example

```
<%
Option Explicit
Dim db, stats
Set db = Server.CreateObject("G3DB")
```

```
If db.Open("mysql", "user:pass@tcp(localhost)/db") Then
  Set stats = db.Stats()
  If IsObject(stats) Then Response.Write stats("OpenConnections")
  db.Close
End If

Set db = Nothing
%>
```

171.8 API Reference

- **Object**: G3DB
- **Method**: Stats
- **Arguments**: none
- **Returns**: Scripting.Dictionary on open connection, Empty otherwise

172 Driver Property

172.1 Overview

Gets or sets the normalized database driver name.

172.2 Prerequisites

```
Set db = Server.CreateObject("G3DB")
```

172.3 Syntax

```
db.Driver = "mysql"
driverName = db.Driver
```

172.4 Return Value

Returns a **String** with the current normalized driver value.

172.5 Remarks

- This property is read/write.
- Assigned values are normalized to canonical driver names.

172.6 Example

```
<%
Dim db
Set db = Server.CreateObject("G3DB")

db.Driver = "postgresql"
Response.Write db.Driver
```

```
Set db = Nothing
%>
```

172.7 API Reference

- **Type:** String
- **Access:** Read/write

173 DSN Property

173.1 Overview

Gets or sets the connection string value stored in the G3DB object.

173.2 Prerequisites

```
Set db = Server.CreateObject("G3DB")
```

173.3 Syntax

```
db.DSN = "user:pass@tcp(localhost:3306)/app"
currentDsn = db.DSN
```

173.4 Return Value

Returns a **String** with the current DSN value.

173.5 Remarks

- This property is read/write.
- `Open(driver, dsn)` also updates this property when open succeeds.

173.6 Example

```
<%
Dim db
Set db = Server.CreateObject("G3DB")

db.DSN = "data.db"
Response.Write db.DSN

Set db = Nothing
%>
```

173.7 API Reference

- **Type:** String
- **Access:** Read/write

174 IsOpen Property

174.1 Overview

Reports whether the database connection pool is currently open.

174.2 Prerequisites

```
Set db = Server.CreateObject("G3DB")
```

174.3 Syntax

```
status = db.IsOpen
```

174.4 Return Value

Returns a **Boolean**: - **True** when connection is open. - **False** when connection is closed or not initialized.

174.5 Remarks

- This property is read-only.

174.6 Example

```
<%
Dim db
Set db = Server.CreateObject("G3DB")

Response.Write db.IsOpen

Set db = Nothing
%>
```

174.7 API Reference

- **Type:** Boolean
- **Access:** Read-only

175 LastError Property

175.1 Overview

Returns the latest error message recorded by the G3DB object.

175.2 Prerequisites

```
Set db = Server.CreateObject("G3DB")
```

175.3 Syntax

```
errText = db.LastError
```

175.4 Return Value

Returns a **String**: - Error text when an operation fails. - Empty string when no error is currently stored.

175.5 Remarks

- This property is read-only.
- Same value is returned by `GetError()`.

175.6 Example

```
<%
Dim db
Set db = Server.CreateObject("G3DB")

If Not db.Open("mysql", "bad") Then
    Response.Write db.LastError
End If

Set db = Nothing
%>
```

175.7 API Reference

- **Type:** String
- **Access:** Read-only

176 Use the G3HTTP Library

176.1 Overview

Use **G3HTTP** to execute outbound HTTP requests from G3Pix AxonASP scripts. The library supports configurable HTTP methods, optional request bodies, and automatic JSON parsing when the response content type is JSON.

176.2 Prerequisites

- Outbound network access to the destination host.
- Valid TLS trust configuration for HTTPS endpoints.
- Create the object with the primary ProgID:

```
Set http = Server.CreateObject("G3HTTP")
```

176.3 How It Works

- `Fetch` and `Request` share the same implementation.
- Default HTTP method is `GET` when not provided.
- When a request body is provided, `Content-Type` is set to `application/json`.

- Requests use a 10-second timeout.
- If response `Content-Type` contains `application/json`, the body is parsed through G3JSON.
- If JSON parsing fails, the raw response body is returned as text.

176.4 API Reference

176.4.1 Methods

- **Fetch(url[, method][, body])**: Sends an HTTP request and returns parsed JSON or text.
- **Request(url[, method][, body])**: Alias of `Fetch`.

176.4.2 Properties

G3HTTP does not expose public properties.

176.5 Example

```
<%
Dim http, result
Set http = Server.CreateObject("G3HTTP")

result = http.Fetch("https://api.example.com/status")

If IsObject(result) Then
    Response.Write result("status")
ElseIf IsArray(result) Then
    Response.Write "Array response"
ElseIf Not IsEmpty(result) Then
    Response.Write result
Else
    Response.Write "Request failed"
End If

Set http = Nothing
%>
```

177 G3HTTP Methods

177.1 Overview

This page summarizes every method exposed by `G3HTTP` in G3Pix AxonASP.

177.2 Methods

Method	Returns	Description
`Fetch(url [, method] [, body])`	Dictionary, Array, scalar, String, or Empty	Sends an outbound HTTP request. Returns parsed

Method	Returns	Description
		JSON-native values when JSON parsing succeeds, raw response String when response is non-JSON or JSON parse fails, and Empty on request/IO failures.
`Request(url [, method] [, body])`	Dictionary, Array, scalar, String, or Empty	Alias of `Fetch` with the same runtime behavior and return contract.

177.3 Remarks

- Instantiate the library with `Server.CreateObject("G3HTTP")`.
- Method names are case-insensitive.
- Default timeout is 10 seconds per request.

178 G3HTTP Properties

178.1 Overview

This page lists properties exposed by `G3HTTP`.

178.2 Properties

Property	Access	Type	Description
None	N/A	N/A	`G3HTTP` does not expose public properties. Use `Fetch` and `Request` methods.

178.3 Remarks

- Instantiate the library with `Server.CreateObject("G3HTTP")`.

179 Send an HTTP Request with Fetch

179.1 Overview

Executes an outbound HTTP request and returns parsed JSON content or raw response text.

179.2 Prerequisites

Instantiate the library with `Server.CreateObject("G3HTTP")`.

179.3 Syntax

```
result = http.Fetch(url[, method][, body])
```

179.4 Parameters

Parameter	Type	Required	Description
url	String	Yes	Absolute request URL.
method	String	No	HTTP method. Default is `GET`.
body	String	No	Request payload. When present, `Content-Type` is set to `application/json`.

179.5 Return Value

- **Scripting.Dictionary**: Returned when response content type is JSON and the root is an object.
- **Array**: Returned when response content type is JSON and the root is an array.
- **String** / **Integer** / **Double** / **Boolean** / **Null**: Returned when response content type is JSON and the root is a primitive that parses successfully.
- **String**: Raw response body for non-JSON responses.
- **String**: Raw response body when content type is JSON but parsing fails.
- **Empty**: Returned when URL is missing, request creation fails, request execution fails, or response read fails.

179.6 Remarks

- Request timeout is 10 seconds.
- Method names are case-insensitive.

179.7 Example

```
<%
Option Explicit
Dim http, result, payload
Set http = Server.CreateObject("G3HTTP")

payload = "{""id"": 10}"
result = http.Fetch("https://api.example.com/items", "POST", payload)
```

```
If IsObject(result) Then
    Response.Write result("status")
ElseIf Not IsEmpty(result) Then
    Response.Write result
Else
    Response.Write "Request failed"
End If

Set http = Nothing
%>
```

179.8 API Reference

- **Object**: `G3HTTP`
- **Method**: `Fetch`
- **Arguments**: `url` (String, required), `method` (String, optional), `body` (String, optional)
- **Returns**: Dictionary, Array, scalar primitive, raw response String, or Empty on request failure

180 Use the G3MAIL Library

180.1 Overview

The **G3MAIL** library provides high-performance SMTP (Simple Mail Transfer Protocol) capabilities for G3Pix AxonASP applications. It enables the delivery of electronic mail messages with support for plain text, HTML formatting, multiple recipients (To, CC, BCC), and secure authentication. The library is designed for zero-allocation performance and can leverage system environment variables for default SMTP configuration.

180.2 Syntax

To instantiate the library, use the following syntax:

```
Set mail = Server.CreateObject("G3MAIL")
```

180.3 Prerequisites

- **SMTP Server access**: Requires a valid SMTP server hostname or IP address.
- **Authentication**: Valid credentials (Username and Password) are typically required for external relaying.
- **Network access**: The server hosting AxonASP must have outbound access to the SMTP port (usually 25, 465, or 587).

180.4 How it Works

The G3MAIL object operates as a stateful message builder. You configure the server connection details and message properties (such as **Subject**, **Body**, and recipients) before calling the **Send** method.

If SMTP properties are not explicitly set in the script, the library automatically attempts to retrieve configuration from the following environment variables: - `SMTP_HOST` - `SMTP_PORT` - `SMTP_USER` - `SMTP_PASS` - `SMTP_FROM`

180.5 API Reference

180.5.1 Methods

- **AddAddress**: Appends a recipient to the primary destination list.
- **AddBcc**: Appends a recipient to the blind carbon copy list.
- **AddCc**: Appends a recipient to the carbon copy list.
- **Clear**: Resets all message fields and recipient lists to their default state.
- **Send**: Connects to the SMTP server and delivers the message.

180.5.2 Properties

- **Body**: Sets the message content. Automatically detects format based on **IsHTML**.
- **BodyFormat**: Sets the message format (0 for HTML, 1 for Text).
- **From**: Sets the sender's email address.
- **FromName**: Sets the display name for the sender.
- **Host**: Sets the SMTP server address.
- **IsHTML**: Specifies whether the body should be treated as HTML.
- **Password**: Sets the authentication password.
- **Port**: Sets the SMTP server port.
- **Subject**: Sets the message subject line.
- **To**: Sets the primary recipient list (comma or semicolon separated).
- **Username**: Sets the authentication username.

180.6 Code Example

The following example demonstrates how to configure and send an HTML email.

```
<%
Dim mail, result
Set mail = Server.CreateObject("G3MAIL")

' Configure SMTP Server
mail.Host = "smtp.example.com"
mail.Port = 587
```

```
mail.Username = "user@example.com"
mail.Password = "securePassword123"

' Configure Message
mail.From = "noreply@example.com"
mail.FromName = "AxonASP Notification"
mail.Subject = "System Alert: Update Completed"
mail.HTMLBody = "<h1>Update Successful</h1><p>Your system has been update
d to the latest version.</p>"

' Add Recipients
mail.AddAddress "admin@example.com"
mail.AddCc "backup@example.com"

' Send Email
result = mail.Send()

If result = True Then
    Response.Write "Message delivered successfully."
Else
    Response.Write "Delivery failed: " & result
End If

Set mail = Nothing
%>
```

181 G3MAIL Methods

181.1 Overview

This page summarizes every method exposed by `G3MAIL` in G3Pix AxonASP.

181.2 Methods

Method	Returns	Description
`AddAddress(address)`	Boolean	Adds one recipient address to the To list. Returns `True`.
`AddCC(address)`	Boolean	Adds one recipient address to the CC list. Returns `True`.
`AddBCC(address)`	Boolean	Adds one recipient address to the BCC list. Returns `True`.
`Clear()`	Boolean	Clears To/CC/BCC/Subject/Body

Method	Returns	Description
		context and resets HTML mode. Returns `True`.
`Send()`	Boolean or String	Sends using configured properties and returns `True` on success; returns an error String when SMTP configuration is incomplete or send fails.
`Send(to, subject, body)`	Boolean or String	CDONTS-style overload that first sets To/Subject/Body, then sends. Returns `True` on success; returns an error String on failure.

181.3 Remarks

- Instantiate the library with `Server.CreateObject("G3MAIL")`.
- Method names are case-insensitive.
- `Send` does not return Empty for operational failure; it returns an error string.

182 G3MAIL Properties

182.1 Overview

This page lists the properties exposed by `G3MAIL`.

182.2 Properties

Property	Access	Type	Description
`Host`	Read/Write	String	SMTP host name.
`Port`	Read/Write	Integer	SMTP port number.
`Username`	Read/Write	String	SMTP authentication username.
`Password`	Read/Write	String	SMTP authentication password.
`From`	Read/Write	String	Sender email address used in the message header.

Property	Access	Type	Description
`FromName`	Read/Write	String	Display name for the sender.
`To`	Read/Write	String	Recipient list for To, represented as comma-separated text.
`CC`	Read/Write	String	Recipient list for CC, represented as comma-separated text.
`BCC`	Read/Write	String	Recipient list for BCC, represented as comma-separated text.
`Subject`	Read/Write	String	Message subject line.
`Body`	Read/Write	String	Message body text.
`IsHTML`	Read/Write	Boolean	Toggles HTML (`True`) or plain-text (`False`) body mode.
`BodyFormat`	Read/Write	Integer	Body format selector (`0` for HTML, `1` for plain text).

182.3 Remarks

- Instantiate the library with `Server.CreateObject("G3MAIL")`.
- Property names are case-insensitive.

183 AddAddress Method

183.1 Overview

The **AddAddress** method appends a recipient email address to the primary destination list (To) of the G3Pix AxonASP G3MAIL object.

183.2 Syntax

```
result = mail.AddAddress(email)
```

183.3 Parameters and Arguments

- **email** (String, Required): The recipient's email address (e.g., "user@example.com").

183.4 Return Values

Returns a **Boolean** value (True).

183.5 Remarks

- You can call this method multiple times to add multiple recipients to the **To** list.
- Invalid email formats are not validated by this method; validation occurs during the **Send** operation.
- The method is case-insensitive and can also be called as **AddRecipient** or **AddTo**.

183.6 Code Example

```
<%
Dim mail
Set mail = Server.CreateObject("G3MAIL")

' Add multiple recipients
mail.AddAddress "admin@example.com"
mail.AddAddress "support@example.com"

' Alternatively using the property
mail.To = "user1@example.com, user2@example.com"

Set mail = Nothing
%>
```

184 AddBcc Method

184.1 Overview

The **AddBcc** method appends a recipient email address to the blind carbon copy (BCC) list of the G3Pix AxonASP G3MAIL object.

184.2 Syntax

```
result = mail.AddBcc(email)
```

184.3 Parameters and Arguments

- **email** (String, Required): The recipient's email address to be BCC'd.

184.4 Return Values

Returns a **Boolean** value (True).

184.5 Remarks

- Recipients added via this method are not visible to other recipients of the email.
- Use this method for privacy when sending to multiple recipients who should not see each other's addresses.

184.6 Code Example

```
<%
Dim mail
Set mail = Server.CreateObject("G3MAIL")

mail.Subject = "Newsletter"
mail.AddAddress "subscriber@example.com"
mail.AddBcc "archive@example.com"
mail.AddBcc "compliance@example.com"

Set mail = Nothing
%>
```

185 AddCc Method

185.1 Overview

The **AddCc** method appends a recipient email address to the carbon copy (CC) list of the G3Pix AxonASP G3MAIL object.

185.2 Syntax

```
result = mail.AddCc(email)
```

185.3 Parameters and Arguments

- **email** (String, Required): The recipient's email address to be CC'd.

185.4 Return Values

Returns a **Boolean** value (True).

185.5 Remarks

- Recipients added via this method will be visible to all other recipients of the email.
- You can call this method multiple times to add multiple CC recipients.

185.6 Code Example

```
<%
Dim mail
Set mail = Server.CreateObject("G3MAIL")

mail.Subject = "Project Update"
mail.AddAddress "manager@example.com"
mail.AddCc "team-lead@example.com"
mail.AddCc "backup@example.com"

Set mail = Nothing
%>
```

186 Clear Method

186.1 Overview

The **Clear** method resets all message properties and recipient lists of the G3Pix AxonASP G3MAIL object to their default empty states.

186.2 Syntax

```
result = mail.Clear()
```

186.3 Parameters and Arguments

This method does not take any arguments.

186.4 Return Values

Returns a **Boolean** value (True).

186.5 Remarks

- This method clears the **To**, **Cc**, and **Bcc** lists, as well as the **Subject**, **Body**, and **IsHTML** status.
- Use this method when reusing the same G3MAIL object to send multiple different emails in a single script execution.
- SMTP server settings (Host, Port, Username, Password) are NOT cleared by this method.

186.6 Code Example

```
<%
Dim mail
Set mail = Server.CreateObject("G3MAIL")

' First email
mail.AddAddress "user1@example.com"
```

```
mail.Subject = "First Message"
mail.Body = "Hello User 1"
mail.Send()

' Clear for second email
mail.Clear()

' Second email
mail.AddAddress "user2@example.com"
mail.Subject = "Second Message"
mail.Body = "Hello User 2"
mail.Send()

Set mail = Nothing
%>
```

187 Send Method

187.1 Overview

The **Send** method attempts to deliver the email message using the configured SMTP server and message properties for the G3Pix AxonASP G3MAIL object.

187.2 Syntax

```
result = mail.Send([to, subject, body])
```

187.3 Parameters and Arguments

- **to** (String, Optional): The recipient email address. If provided, it overrides the **To** property for this call.
- **subject** (String, Optional): The message subject. If provided, it overrides the **Subject** property for this call.
- **body** (String, Optional): The message body. If provided, it overrides the **Body** property for this call and sets the format to plain text.

187.4 Return Values

Returns a **Boolean** value (True) if the email was sent successfully. If the operation fails, it returns a **String** containing the error message.

187.5 Remarks

- If SMTP server properties (Host, Port, etc.) are not set, the library will attempt to use environment variables (`SMTP_HOST`, `SMTP_PORT`, `SMTP_USER`, `SMTP_PASS`, `SMTP_FROM`).
- At least one recipient must be defined (via **AddAddress**, **AddCc**, **AddBcc**, or the **To** property).

- The method uses the `gomail` library internally for reliable delivery.

187.6 Code Example

```
<%
Dim mail, result
Set mail = Server.CreateObject("G3MAIL")

mail.Host = "smtp.example.com"
mail.From = "system@example.com"
mail.Subject = "Test Email"
mail.Body = "This is a test message."
mail.AddAddress "recipient@example.com"

result = mail.Send()

If result = True Then
    Response.Write "Success!"
Else
    Response.Write "Error: " & result
End If

Set mail = Nothing
%>
```

188 Bcc Property

188.1 Overview

The **Bcc** property gets or sets the blind carbon copy (BCC) recipient list for the G3Pix AxonASP G3MAIL object.

188.2 Syntax

```
value = mail.Bcc
mail.Bcc = newValue
```

188.3 Parameters and Arguments

- **newValue** (String): A comma-separated or semicolon-separated list of email addresses.

188.4 Return Values

Returns a **String** containing the current BCC recipients, separated by commas.

188.5 Remarks

- Recipients in the BCC list are hidden from other recipients.

- Setting this property replaces any existing BCC recipients. Use the **AddBcc** method to append to the list instead.

188.6 Code Example

```
<%
Dim mail
Set mail = Server.CreateObject("G3MAIL")

mail.Bcc = "audit@example.com, backup@example.com"
Response.Write "BCC List: " & mail.Bcc

Set mail = Nothing
%>
```

189 Body Property

189.1 Overview

The **Body** property gets or sets the main content of the email message for the G3Pix AxonASP G3MAIL object.

189.2 Syntax

```
value = mail.Body
mail.Body = newValue
```

189.3 Parameters and Arguments

- **newValue** (String): The text or HTML content of the email.

189.4 Return Values

Returns a **String** containing the current message body.

189.5 Remarks

- Setting this property via **Body**, **Message**, or **TextBody** automatically sets **IsHTML** to False.
- Setting this property via **HtmlBody** automatically sets **IsHTML** to True.
- This property is essential for the **Send** method to have content to deliver.

189.6 Code Example

```
<%
Dim mail
Set mail = Server.CreateObject("G3MAIL")

mail.Body = "Hello, this is a plain text message."
' To send HTML:
```

```
' mail.HtmlBody = "<h1>Hello</h1><p>This is HTML.</p>"

Set mail = Nothing
%>
```

190 BodyFormat Property

190.1 Overview

The **BodyFormat** property gets or sets the format of the email message body (Plain Text or HTML) for the G3Pix AxonASP G3MAIL object.

190.2 Syntax

```
value = mail.BodyFormat
mail.BodyFormat = newValue
```

190.3 Parameters and Arguments

- **newValue** (Integer): Use 0 for HTML format or 1 for Plain Text format.

190.4 Return Values

Returns an **Integer**: 0 if the message is in HTML format, 1 if it is in Plain Text format.

190.5 Remarks

- This property is synchronized with the **IsHTML** property.
- Setting **BodyFormat** to 0 sets **IsHTML** to True. Setting it to 1 (or any other value) sets **IsHTML** to False.
- This property can also be accessed using the alias **MailFormat**.

190.6 Code Example

```
<%
Dim mail
Set mail = Server.CreateObject("G3MAIL")

mail.BodyFormat = 0 ' Set to HTML
If mail.BodyFormat = 0 Then
    Response.Write "Message is in HTML format."
End If

Set mail = Nothing
%>
```

191 Cc Property

191.1 Overview

The **Cc** property gets or sets the carbon copy (CC) recipient list for the G3Pix AxonASP G3MAIL object.

191.2 Syntax

```
value = mail.Cc
mail.Cc = newValue
```

191.3 Parameters and Arguments

- **newValue** (String): A comma-separated or semicolon-separated list of email addresses.

191.4 Return Values

Returns a **String** containing the current CC recipients, separated by commas.

191.5 Remarks

- Recipients in the CC list are visible to all other recipients.
- Setting this property replaces any existing CC recipients. Use the **AddCc** method to append to the list instead.

191.6 Code Example

```
<%
Dim mail
Set mail = Server.CreateObject("G3MAIL")

mail.Cc = "cc1@example.com, cc2@example.com"
Response.Write "CC List: " & mail.Cc

Set mail = Nothing
%>
```

192 From Property

192.1 Overview

The **From** property gets or sets the sender's email address for the G3Pix AxonASP G3MAIL object.

192.2 Syntax

```
value = mail.From
mail.From = newValue
```

192.3 Parameters and Arguments

- **newValue** (String): The email address of the sender.

192.4 Return Values

Returns a **String** containing the sender's email address.

192.5 Remarks

- This property is required for sending emails unless the `SMTP_FROM` environment variable is set.
- To set a display name (e.g., "Company Name sender@example.com"), use the **FromName** property.
- This property can also be accessed using the alias **FromAddress**.

192.6 Code Example

```
<%
Dim mail
Set mail = Server.CreateObject("G3MAIL")

mail.From = "no-reply@example.com"
Response.Write "Sender: " & mail.From

Set mail = Nothing
%>
```

193 FromName Property

193.1 Overview

The **FromName** property gets or sets the display name of the sender for the G3Pix AxonASP G3MAIL object.

193.2 Syntax

```
value = mail.FromName
mail.FromName = newValue
```

193.3 Parameters and Arguments

- **newValue** (String): The friendly display name of the sender.

193.4 Return Values

Returns a **String** containing the sender's display name.

193.5 Remarks

- When set, the recipient will see this name in their mail client instead of just the email address.
- This property works in conjunction with the **From** property.

193.6 Code Example

```
<%
Dim mail
Set mail = Server.CreateObject("G3MAIL")

mail.From = "info@g3pix.com.br"
mail.FromName = "G3Pix Support"

Set mail = Nothing
%>
```

194 Host Property

194.1 Overview

The **Host** property gets or sets the SMTP server hostname or IP address for the G3Pix AxonASP G3MAIL object.

194.2 Syntax

```
value = mail.Host
mail.Host = newValue
```

194.3 Parameters and Arguments

- **newValue** (String): The address of the SMTP server (e.g., “smtp.gmail.com”).

194.4 Return Values

Returns a **String** containing the current SMTP host.

194.5 Remarks

- If not set, the library attempts to use the `SMTP_HOST` environment variable.
- This property can also be accessed using the aliases **MailHost** or **SMTPServer**.

194.6 Code Example

```
<%
Dim mail
Set mail = Server.CreateObject("G3MAIL")

mail.Host = "smtp.office365.com"
Response.Write "Current Host: " & mail.Host
```

```
Set mail = Nothing
%>
```

195 IsHTML Property

195.1 Overview

The **IsHTML** property gets or sets a value indicating whether the email body should be treated as HTML for the G3Pix AxonASP G3MAIL object.

195.2 Syntax

```
value = mail.IsHTML
mail.IsHTML = newValue
```

195.3 Parameters and Arguments

- **newValue** (Boolean): True for HTML content, False for plain text.

195.4 Return Values

Returns a **Boolean** value.

195.5 Remarks

- Setting this property directly affects how the **Body** content is rendered by the recipient's mail client.
- This property is synchronized with **BodyFormat**.
- Some aliases like **HtmlBody** set this property to True automatically.

195.6 Code Example

```
<%
Dim mail
Set mail = Server.CreateObject("G3MAIL")

mail.Body = "<h1>Notice</h1><p>System maintenance tonight.</p>"
mail.IsHTML = True

Set mail = Nothing
%>
```

196 Password Property

196.1 Overview

The **Password** property sets the authentication password for the SMTP server for the G3Pix AxonASP G3MAIL object.

196.2 Syntax

```
value = mail.Password
mail.Password = newValue
```

196.3 Parameters and Arguments

- **newValue** (String): The password for the SMTP account.

196.4 Return Values

Returns a **String** containing the current password (if set).

196.5 Remarks

- For security reasons, avoid hardcoding passwords in scripts. Use environment variables (`SMTP_PASS`) or encrypted configuration files.
- This property can also be accessed using the aliases **Pass** or **AuthPassword**.

196.6 Code Example

```
<%
Dim mail
Set mail = Server.CreateObject("G3MAIL")

mail.Password = "mySecretPassword"

Set mail = Nothing
%>
```

197 Port Property

197.1 Overview

The **Port** property gets or sets the port number for the SMTP server for the G3Pix AxonASP G3MAIL object.

197.2 Syntax

```
value = mail.Port
mail.Port = newValue
```

197.3 Parameters and Arguments

- **newValue** (Integer): The port number (typically 25, 465, or 587).

197.4 Return Values

Returns an **Integer** representing the current port number.

197.5 Remarks

- If not set, the library attempts to use the `SMTP_PORT` environment variable.
- This property can also be accessed using the alias **SMTPServerPort**.

197.6 Code Example

```
<%
Dim mail
Set mail = Server.CreateObject("G3MAIL")

mail.Port = 587
Response.Write "SMTP Port: " & mail.Port

Set mail = Nothing
%>
```

198 Subject Property

198.1 Overview

The **Subject** property gets or sets the subject line of the email for the G3Pix AxonASP G3MAIL object.

198.2 Syntax

```
value = mail.Subject
mail.Subject = newValue
```

198.3 Parameters and Arguments

- **newValue** (String): The text for the email subject.

198.4 Return Values

Returns a **String** containing the current subject.

198.5 Remarks

- The subject line should be concise and relevant to the message content.
- It is highly recommended to set a subject for every email to avoid being flagged as spam.

198.6 Code Example

```
<%
Dim mail
Set mail = Server.CreateObject("G3MAIL")

mail.Subject = "Urgent: System Report"
Response.Write "Subject: " & mail.Subject
```

```
Set mail = Nothing
%>
```

199 To Property

199.1 Overview

The **To** property gets or sets the primary recipient list for the G3Pix AxonASP G3MAIL object.

199.2 Syntax

```
value = mail.To
mail.To = newValue
```

199.3 Parameters and Arguments

- **newValue** (String): A comma-separated or semicolon-separated list of email addresses.

199.4 Return Values

Returns a **String** containing the current primary recipients, separated by commas.

199.5 Remarks

- At least one recipient must be specified (either here or via **AddAddress**) before calling **Send**.
- Setting this property replaces any existing primary recipients. Use the **AddAddress** method to append to the list instead.

199.6 Code Example

```
<%
Dim mail
Set mail = Server.CreateObject("G3MAIL")

mail.To = "customer1@example.com, customer2@example.com"
Response.Write "Recipients: " & mail.To

Set mail = Nothing
%>
```

200 Username Property

200.1 Overview

The **Username** property gets or sets the authentication username for the SMTP server for the G3Pix AxonASP G3MAIL object.

200.2 Syntax

```
value = mail.Username
mail.Username = newValue
```

200.3 Parameters and Arguments

- **newValue** (String): The username or email address for SMTP authentication.

200.4 Return Values

Returns a **String** containing the current username.

200.5 Remarks

- If not set, the library attempts to use the `SMTP_USER` environment variable.
- This property can also be accessed using the aliases **User** or **AuthUsername**.

200.6 Code Example

```
<%
Dim mail
Set mail = Server.CreateObject("G3MAIL")

mail.Username = "webmaster@example.com"
Response.Write "Auth User: " & mail.Username

Set mail = Nothing
%>
```

201 Use the G3IMAGE Library

201.1 Overview

The **G3IMAGE** library is a high-performance native image processing and 2D drawing engine for G3Pix AxonASP. It enables developers to programmatically create, load, manipulate, and render images directly on the server. The library supports standard formats including PNG and JPEG, and provides a rich set of drawing primitives such as lines, rectangles, circles, and high-quality text rendering with TrueType font support.

201.2 Syntax

To instantiate the library, use the following syntax:

```
Set image = Server.CreateObject("G3IMAGE")
```

201.3 Prerequisites

- **Memory**: Ensure sufficient server memory for high-resolution image processing.
- **File System Permissions**: Requires read/write access to the directories where images are loaded or saved.
- **Fonts**: TrueType (.ttf) or OpenType (.otf) font files are required for advanced text rendering.

201.4 How it Works

The G3IMAGE object operates by managing a drawing context. You can either create a new blank context using **NewContext** or load an existing image using **LoadImage**. Once a context is established, you can perform drawing operations like **DrawLine**, **DrawRectangle**, and **DrawString**.

After completing the drawing operations, the image can be saved to disk using **SavePNG** or **SaveJPG**, or rendered directly to a byte array using **RenderViaTemp** for immediate delivery to the client via `Response.BinaryWrite`.

The library also supports advanced features like anti-aliasing, alpha transparency (PNG), and configurable JPEG compression quality.

201.5 API Reference

201.5.1 Methods

- **Close**: Releases all internal resources and clears the active context.
- **DrawCircle**: Draws a circle at the specified coordinates.
- **DrawEllipse**: Draws an ellipse at the specified coordinates.
- **DrawImage**: Draws a previously loaded image onto the current canvas.
- **DrawLine**: Draws a straight line between two points.
- **DrawRectangle**: Draws a rectangle with the specified dimensions.
- **DrawString**: Renders text at the specified coordinates.
- **DrawStringAnchored**: Renders text with precise anchor alignment.
- **Fill**: Fills the current path with the active color.
- **LoadFontFace**: Loads a TrueType font for text rendering.
- **LoadImage**: Loads an image file into memory.
- **LoadJPG**: Explicitly loads a JPEG image file.
- **LoadPNG**: Explicitly loads a PNG image file.
- **MeasureString**: Returns the width and height of a text string in pixels.
- **NewContext**: Initializes a new drawing canvas with the specified width and height.
- **RenderViaTemp**: Renders the current image to a byte array for binary output.

- **SaveJPG**: Saves the current canvas to a JPEG file.
- **SavePNG**: Saves the current canvas to a PNG file.
- **SetColor**: Sets the active drawing color using RGB or RGBA values.
- **SetHexColor**: Sets the active drawing color using a hexadecimal string.
- **SetLineWidth**: Configures the thickness of lines and strokes.
- **Stroke**: Outlines the current path with the active color.

201.5.2 Properties

- **DefaultFormat**: Gets or sets the preferred output format (png or jpg).
- **HasContext**: Indicates whether a drawing canvas is currently initialized.
- **Height**: Returns the height of the current canvas in pixels.
- **JPGQuality**: Gets or sets the compression quality for JPEG output (1-100).
- **LastBytes**: Returns the raw byte array from the last render operation.
- **LastError**: Returns the description of the most recent error.
- **LastMimeType**: Returns the MIME type of the last rendered image.
- **LastTempFile**: Returns the path to the temporary file used during rendering.
- **Width**: Returns the width of the current canvas in pixels.

201.6 Code Example

The following example demonstrates how to create a simple image with text and output it to the browser.

```
<%
Dim img, bytes
Set img = Server.CreateObject("G3IMAGE")

' Create a 400x200 canvas
If img.NewContext(400, 200) Then
    ' Fill background with white
    img.SetHexColor "#FFFFFF"
    img.Clear

    ' Draw a blue rectangle
    img.SetHexColor "#003399"
    img.DrawRectangle 10, 10, 380, 180
    img.SetLineWidth 2
    img.Stroke

    ' Draw some text
    img.SetHexColor "#333333"
    ' Note: Requires a valid font path
    If img.LoadFontFace("C:\Windows\Fonts\arial.ttf", 24) Then
        img.DrawStringAnchored "AxonASP G3IMAGE", 200, 100, 0.5, 0.5
    End If
```

```
    ' Render to byte array as PNG
    bytes = img.RenderViaTemp("png")

    ' Output to browser
    Response.ContentType = img.LastMimeType
    Response.BinaryWrite bytes
Else
    Response.Write "Error: " & img.LastError
End If

Set img = Nothing
%>
```

202 G3IMAGE Methods

202.1 Overview

This page summarizes every method exposed by `G3IMAGE` in G3Pix AxonASP.

202.2 Methods

Method	Returns	Description
`Close()`	Boolean	Releases drawing/image resources and resets internal state. Returns `True`.
`NewContext(width, height)`	Boolean or Empty	Creates a new drawing context. Returns `True` on success, Empty when arguments are invalid.
`LoadImage(path)`	Boolean or Empty	Loads image from file using auto-decoding. Returns `True` on success, Empty on failure.
`LoadPNG(path)`	Boolean or Empty	Loads PNG image from file. Returns `True` on success, Empty on failure.
`LoadJPG(path)`	Boolean or Empty	Loads JPEG image from file. Returns `True` on success, Empty on failure.
`ContextForImage()`	Boolean	Creates drawing context from last loaded image. Returns `True` on success,

Method	Returns	Description
		`False` when no image is loaded.
`SavePNG(path)`	Boolean	Saves active context as PNG. Returns `True` on success; otherwise `False`.
`SaveJPG(path [, quality])`	Boolean	Saves active context as JPEG. Returns `True` on success; otherwise `False`.
`SetHexColor(hexColor)`	Empty	Sets current drawing color from hexadecimal value.
`SetColor(colorText)`	Empty	Sets current drawing color from color text value.
`Clear()`	Empty	Clears the active context.
`SetLineWidth(width)`	Empty	Sets stroke width for subsequent path operations.
`DrawLine(x1, y1, x2, y2)`	Empty	Adds a line path segment to the active context.
`DrawRectangle(x, y, width, height)`	Empty	Adds a rectangle path to the active context.
`DrawCircle(x, y, radius)`	Empty	Adds a circle path to the active context.
`DrawEllipse(x, y, rx, ry)`	Empty	Adds an ellipse path to the active context.
`Stroke()`	Empty	Strokes the current path.
`Fill()`	Empty	Fills the current path.
`FillPreserve()`	Empty	Fills the current path and preserves path data.
`StrokePreserve()`	Empty	Strokes the current path and preserves path data.
`LoadFontFace(path, points)`	Boolean	Loads a font face and applies it to active context when present. Returns `True` on success; otherwise `False`.
`DrawString(text, x, y)`	Empty	Draws text at coordinates.

Method	Returns	Description
`DrawStringAnchored(text, x, y, ax, ay)`	Empty	Draws anchored text with alignment factors.
`MeasureString(text)`	Array or Empty	Returns two-element array `[width, height]` as Double values, or Empty when no context is active.
`DrawImage(x, y)`	Empty	Draws last loaded image at destination coordinates in the active context.
`RenderViaTemp([format][, quality])`	Array or Empty	Renders current image and returns byte array; returns Empty on render failure.

202.3 Remarks

- Instantiate the library with `Server.CreateObject("G3IMAGE")`.
- Method names are case-insensitive.
- Inspect `LastError` when methods return `False` or Empty.

203 G3IMAGE Properties

203.1 Overview

This page lists the properties exposed by `G3IMAGE`.

203.2 Properties

Property	Access	Type	Description
`HasContext`	Read-only	Boolean	Indicates whether an active drawing context exists.
`Width`	Read-only	Integer	Pixel width of the active drawing context, or `0` when no context exists.
`Height`	Read-only	Integer	Pixel height of the active drawing context, or `0` when no context exists.
`LastError`	Read-only	String	Latest operation error message.

Property	Access	Type	Description
`LastMimeType`	Read-only	String	MIME type of the most recent rendered output.
`LastTempFile`	Read-only	String	Temporary file path used by the most recent temp render operation.
`LastBytes`	Read-only	Array	Byte array from the most recent render operation.
`DefaultFormat`	Read/Write	String	Default render format (`png`, `jpg`, or `jpeg`).
`JPGQuality`	Read/Write	Integer	JPEG quality setting clamped to the range `1..100`.
`AlignLeft`	Read-only	Integer	Drawing constant for left text alignment.
`AlignCenter`	Read-only	Integer	Drawing constant for centered text alignment.
`AlignRight`	Read-only	Integer	Drawing constant for right text alignment.
`FillRuleWinding`	Read-only	Integer	Drawing constant for winding fill rule.
`FillRuleEvenOdd`	Read-only	Integer	Drawing constant for even-odd fill rule.
`LineCapRound`	Read-only	Integer	Drawing constant for round line caps.
`LineCapButt`	Read-only	Integer	Drawing constant for butt line caps.
`LineCapSquare`	Read-only	Integer	Drawing constant for square line caps.

Property	Access	Type	Description
`LineJoinRound`	Read-only	Integer	Drawing constant for round line joins.
`LineJoinBevel`	Read-only	Integer	Drawing constant for bevel line joins.

203.3 Remarks

- Instantiate the library with `Server.CreateObject("G3IMAGE")`.
- Property names are case-insensitive.

204 Clear Method

204.1 Overview

Clears the current image context using the current fill color. This operation resets the canvas within the active context of the G3Pix AxonASP G3IMAGE library.

204.2 Syntax

```
obj.Clear()
```

204.3 Return Values

Returns Empty upon completion.

204.4 Remarks

- Ensure a context has been initialized using NewContext before calling this method.
- The method uses the color previously set by SetColor or SetHexColor.

204.5 Code Example

```
<%
Dim img
Set img = Server.CreateObject("G3IMAGE")
If img.NewContext(400, 300) Then
    img.SetHexColor("#FFFFFF")
    img.Clear()
End If
Set img = Nothing
%>
```

205 Close Method

205.1 Overview

Releases the resources associated with the G3IMAGE object and clears the current context. This method is part of the G3Pix AxonASP G3IMAGE library.

205.2 Syntax

```
result = obj.Close()
```

205.3 Return Values

Returns a Boolean indicating whether the resources were successfully released.

205.4 Remarks

- It is good practice to call Close when finished with an image to free memory.
- This method also resets any internal error state.

205.5 Code Example

```
<%
Dim img
Set img = Server.CreateObject("G3IMAGE")
If img.NewContext(100, 100) Then
    ' Perform operations
    img.Close()
End If
Set img = Nothing
%>
```

206 DrawCircle Method

206.1 Overview

Adds a circle to the current path in the G3Pix AxonASP G3IMAGE library.

206.2 Syntax

```
obj.DrawCircle(x, y, r)
```

206.3 Parameters

- **x** (Double): The x-coordinate of the center of the circle.
- **y** (Double): The y-coordinate of the center of the circle.
- **r** (Double): The radius of the circle.

206.4 Return Values

Returns Empty upon completion.

206.5 Remarks

- This method adds the circle to the current path. You must call Stroke or Fill to actually render the circle on the canvas.

206.6 Code Example

```
<%
Dim img
Set img = Server.CreateObject("G3IMAGE")
If img.NewContext(200, 200) Then
    img.SetHexColor("#FF0000")
    img.DrawCircle 100, 100, 50
    img.Stroke()
End If
Set img = Nothing
%>
```

207 DrawEllipse Method

207.1 Overview

Adds an ellipse to the current path in the G3Pix AxonASP G3IMAGE library.

207.2 Syntax

```
obj.DrawEllipse(x, y, rx, ry)
```

207.3 Parameters

- **x** (Double): The x-coordinate of the center of the ellipse.
- **y** (Double): The y-coordinate of the center of the ellipse.
- **rx** (Double): The horizontal radius of the ellipse.
- **ry** (Double): The vertical radius of the ellipse.

207.4 Return Values

Returns Empty upon completion.

207.5 Remarks

- This method adds the ellipse to the current path. You must call Stroke or Fill to actually render it on the canvas.

207.6 Code Example

```
<%
Dim img
Set img = Server.CreateObject("G3IMAGE")
If img.NewContext(200, 200) Then
    img.SetHexColor("#00FF00")
```

```
    img.DrawEllipse 100, 100, 80, 40
    img.Fill()
End If
Set img = Nothing
%>
```

208 DrawImage Method

208.1 Overview

Draws the last loaded image onto the current context in the G3Pix AxonASP G3IMAGE library.

208.2 Syntax

```
obj.DrawImage(x, y)
```

208.3 Parameters

- **x** (Integer): The x-coordinate where the top-left corner of the image will be placed.
- **y** (Integer): The y-coordinate where the top-left corner of the image will be placed.

208.4 Return Values

Returns Empty upon completion.

208.5 Remarks

- You must load an image using LoadImage, LoadPNG, or LoadJPG before calling this method.
- The image is drawn directly onto the active context.

208.6 Code Example

```
<%
Dim img
Set img = Server.CreateObject("G3IMAGE")
If img.LoadImage("assets/logo.png") Then
    If img.NewContext(500, 500) Then
        img.DrawImage 10, 10
    End If
End If
Set img = Nothing
%>
```

209 DrawLine Method

209.1 Overview

Adds a line segment to the current path in the G3Pix AxonASP G3IMAGE library.

209.2 Syntax

```
obj.DrawLine(x1, y1, x2, y2)
```

209.3 Parameters

- **x1** (Double): The x-coordinate of the starting point.
- **y1** (Double): The y-coordinate of the starting point.
- **x2** (Double): The x-coordinate of the ending point.
- **y2** (Double): The y-coordinate of the ending point.

209.4 Return Values

Returns Empty upon completion.

209.5 Remarks

- This method defines a line in the current path. Call Stroke to render the line.

209.6 Code Example

```
<%
Dim img
Set img = Server.CreateObject("G3IMAGE")
If img.NewContext(200, 200) Then
    img.SetHexColor("#000000")
    img.SetLineWidth 2
    img.DrawLine 10, 10, 190, 190
    img.Stroke()
End If
Set img = Nothing
%>
```

210 DrawRectangle Method

210.1 Overview

Adds a rectangle to the current path in the G3Pix AxonASP G3IMAGE library.

210.2 Syntax

```
obj.DrawRectangle(x, y, w, h)
```

210.3 Parameters

- **x** (Double): The x-coordinate of the top-left corner.
- **y** (Double): The y-coordinate of the top-left corner.
- **w** (Double): The width of the rectangle.
- **h** (Double): The height of the rectangle.

210.4 Return Values

Returns Empty upon completion.

210.5 Remarks

- This method adds a rectangle to the current path. Call Stroke or Fill to render it.

210.6 Code Example

```
<%
Dim img
Set img = Server.CreateObject("G3IMAGE")
If img.NewContext(400, 300) Then
    img.SetHexColor("#0000FF")
    img.DrawRectangle 50, 50, 300, 200
    img.Stroke()
End If
Set img = Nothing
%>
```

211 DrawString Method

211.1 Overview

Draws a string of text at the specified coordinates in the G3Pix AxonASP G3IMAGE library.

211.2 Syntax

```
obj.DrawString(s, x, y)
```

211.3 Parameters

- **s** (String): The text to be drawn.
- **x** (Double): The x-coordinate for the text position.
- **y** (Double): The y-coordinate for the text position.

211.4 Return Values

Returns Empty upon completion.

211.5 Remarks

- Use LoadFontFace to set the font and size before drawing strings.
- The coordinates typically refer to the baseline of the text.

211.6 Code Example

```
<%
Dim img
Set img = Server.CreateObject("G3IMAGE")
If img.NewContext(400, 200) Then
    img.LoadFontFace "C:\Windows\Fonts\arial.ttf", 24
    img.SetHexColor("#000000")
    img.DrawString "Hello AxonASP", 50, 100
End If
Set img = Nothing
%>
```

212 DrawStringAnchored Method

212.1 Overview

Draws a string of text with specified anchor points for alignment in the G3Pix AxonASP G3IMAGE library.

212.2 Syntax

```
obj.DrawStringAnchored(s, x, y, ax, ay)
```

212.3 Parameters

- **s** (String): The text to be drawn.
- **x** (Double): The x-coordinate for the anchor position.
- **y** (Double): The y-coordinate for the anchor position.
- **ax** (Double): The horizontal anchor point (0.0 for left, 0.5 for center, 1.0 for right).
- **ay** (Double): The vertical anchor point (0.0 for top, 0.5 for middle, 1.0 for bottom).

212.4 Return Values

Returns Empty upon completion.

212.5 Remarks

- This method allows for precise alignment of text relative to a point.
- Use LoadFontFace to set the font and size before drawing.

212.6 Code Example

```
<%
Dim img
Set img = Server.CreateObject("G3IMAGE")
If img.NewContext(400, 200) Then
    img.LoadFontFace "C:\Windows\Fonts\arial.ttf", 24
    img.SetHexColor("#000000")
    ' Draw centered text
    img.DrawStringAnchored "Centered Text", 200, 100, 0.5, 0.5
End If
Set img = Nothing
%>
```

213 Fill Method

213.1 Overview

Fills the current path with the active color in the G3Pix AxonASP G3IMAGE library.

213.2 Syntax

```
obj.Fill()
```

213.3 Return Values

Returns Empty upon completion.

213.4 Remarks

- Use SetColor or SetHexColor to define the fill color before calling this method.
- The path is cleared after the fill operation. Use FillPreserve if you need to keep the path for further operations.

213.5 Code Example

```
<%
Dim img
Set img = Server.CreateObject("G3IMAGE")
If img.NewContext(200, 200) Then
    img.SetHexColor("#FFCC00")
    img.DrawCircle 100, 100, 80
    img.Fill()
End If
Set img = Nothing
%>
```

214 LoadFontFace Method

214.1 Overview

Loads a TrueType font for use in text rendering operations in the G3Pix AxonASP G3IMAGE library.

214.2 Syntax

```
result = obj.LoadFontFace(path, points)
```

214.3 Parameters

- **path** (String): The file path to the .ttf font file.
- **points** (Double): The font size in points.

214.4 Return Values

Returns a Boolean indicating whether the font was successfully loaded.

214.5 Remarks

- If a context is active, this method also sets the font for that context.
- Always check the return value or the LastError property if loading fails.

214.6 Code Example

```
<%
Dim img
Set img = Server.CreateObject("G3IMAGE")
If img.LoadFontFace("C:\Windows\Fonts\verdanab.ttf", 12) Then
    ' Font loaded successfully
End If
Set img = Nothing
%>
```

215 LoadImage Method

215.1 Overview

Loads an image file from the specified path into the G3Pix AxonASP G3IMAGE library.

215.2 Syntax

```
result = obj.LoadImage(path)
```

215.3 Parameters

- **path** (String): The file path to the image. Supports PNG, JPEG, and GIF formats.

215.4 Return Values

Returns a Boolean indicating whether the image was successfully loaded.

215.5 Remarks

- After loading an image, you can use DrawImage to render it into a context.
- The path is resolved relative to the web root if possible.

215.6 Code Example

```
<%
Dim img
Set img = Server.CreateObject("G3IMAGE")
If img.LoadImage("images/header.jpg") Then
    ' Image Loaded
Else
    Response.Write "Error: " & img.LastError
End If
Set img = Nothing
%>
```

216 LoadJPG Method

216.1 Overview

Loads a JPEG image file from the specified path into the G3Pix AxonASP G3IMAGE library.

216.2 Syntax

```
result = obj.LoadJPG(path)
```

216.3 Parameters

- **path** (String): The file path to the JPEG image.

216.4 Return Values

Returns a Boolean indicating whether the image was successfully loaded.

216.5 Remarks

- This method specifically targets JPEG files. For general image loading, use the LoadImage method.
- The path is resolved relative to the web root if possible.

216.6 Code Example

```
<%
Dim img
```

```
Set img = Server.CreateObject("G3IMAGE")
If img.LoadJPG("images/photo.jpg") Then
    ' JPEG loaded successfully
End If
Set img = Nothing
%>
```

217 LoadPNG Method

217.1 Overview

Loads a PNG image file from the specified path into the G3Pix AxonASP G3IMAGE library.

217.2 Syntax

```
result = obj.LoadPNG(path)
```

217.3 Parameters

- **path** (String): The file path to the PNG image.

217.4 Return Values

Returns a Boolean indicating whether the image was successfully loaded.

217.5 Remarks

- This method specifically targets PNG files. For general image loading, use the LoadImage method.
- The path is resolved relative to the web root if possible.

217.6 Code Example

```
<%
Dim img
Set img = Server.CreateObject("G3IMAGE")
If img.LoadPNG("images/logo.png") Then
    ' PNG loaded successfully
End If
Set img = Nothing
%>
```

218 MeasureString Method

218.1 Overview

Measures the width and height of a string if it were rendered with the current font in the G3Pix AxonASP G3IMAGE library.

218.2 Syntax

```
result = obj.MeasureString(s)
```

218.3 Parameters

- **s** (String): The text to measure.

218.4 Return Values

Returns a VBArray containing two Doubles: the width [0] and the height [1] of the string.

218.5 Remarks

- A context must be active and a font must be loaded using LoadFontFace for this method to provide accurate measurements.

218.6 Code Example

```
<%
Dim img, size
Set img = Server.CreateObject("G3IMAGE")
If img.NewContext(400, 200) Then
    img.LoadFontFace "C:\Windows\Fonts\arial.ttf", 12
    size = img.MeasureString("Hello")
    Response.Write "Width: " & size(0) & ", Height: " & size(1)
End If
Set img = Nothing
%>
```

219 NewContext Method

219.1 Overview

Initializes a new drawing context with specified dimensions in the G3Pix AxonASP G3IMAGE library. This method is also accessible via the “New” alias.

219.2 Syntax

```
result = obj.NewContext(Width, Height)
```

219.3 Parameters

- **Width** (Integer): The width of the new context in pixels.
- **Height** (Integer): The height of the new context in pixels.

219.4 Return Values

Returns a Boolean indicating whether the context was successfully created.

219.5 Remarks

- This method maps to both the “New” and “NewContext” method names in the underlying engine.
- Dimensions must be positive integers.
- Calling this method releases any previously active context.

219.6 Code Example

```
<%
Dim img
Set img = Server.CreateObject("G3IMAGE")
' Using the alias "New"
If img.New(800, 600) Then
    ' Context is ready
End If
Set img = Nothing
%>
```

220 RenderViaTemp Method

220.1 Overview

Renders the current context into a temporary file and returns the binary content in the G3Pix AxonASP G3IMAGE library.

220.2 Syntax

```
result = obj.RenderViaTemp(format, quality)
```

220.3 Parameters

- **format** (String): The target format, such as “png” or “jpg”.
- **quality** (Integer): The quality level (1-100), primarily for JPEG output.

220.4 Return Values

Returns a VBArray containing binary bytes. This array is compatible with Response.BinaryWrite.

220.5 Remarks

- This method is useful for streaming the generated image directly to the client browser.
- It handles the creation and cleanup of temporary storage automatically.

220.6 Code Example

```
<%
Dim img, bytes
```

```
Set img = Server.CreateObject("G3IMAGE")
If img.NewContext(100, 100) Then
    img.SetHexColor("#FF0000")
    img.Clear()
    bytes = img.RenderViaTemp("png", 0)
    Response.ContentType = "image/png"
    Response.BinaryWrite bytes
End If
Set img = Nothing
%>
```

221 SaveJPG Method

221.1 Overview

Saves the current context as a JPEG file to the specified path in the G3Pix AxonASP G3IMAGE library.

221.2 Syntax

```
result = obj.SaveJPG(path, quality)
```

221.3 Parameters

- **path** (String): The file path where the JPEG will be saved.
- **quality** (Integer): The quality level (1-100) for the JPEG compression.

221.4 Return Values

Returns a Boolean indicating whether the file was successfully saved.

221.5 Remarks

- Ensure that the application has write permissions for the target directory.
- The path is resolved relative to the web root if possible.

221.6 Code Example

```
<%
Dim img
Set img = Server.CreateObject("G3IMAGE")
If img.NewContext(200, 200) Then
    img.DrawCircle 100, 100, 50
    img.Stroke()
    img.SaveJPG "output.jpg", 85
End If
Set img = Nothing
%>
```

222 SavePNG Method

222.1 Overview

Saves the current context as a PNG file to the specified path in the G3Pix AxonASP G3IMAGE library.

222.2 Syntax

```
result = obj.SavePNG(path)
```

222.3 Parameters

- **path** (String): The file path where the PNG will be saved.

222.4 Return Values

Returns a Boolean indicating whether the file was successfully saved.

222.5 Remarks

- Ensure that the application has write permissions for the target directory.
- The path is resolved relative to the web root if possible.

222.6 Code Example

```
<%
Dim img
Set img = Server.CreateObject("G3IMAGE")
If img.NewContext(200, 200) Then
    img.DrawRectangle 0, 0, 200, 200
    img.Fill()
    img.SavePNG "output.png"
End If
Set img = Nothing
%>
```

223 SetColor Method

223.1 Overview

Sets the active color for drawing and filling operations using an RGB or RGBA string in the G3Pix AxonASP G3IMAGE library.

223.2 Syntax

```
obj.SetColor(rgb_string)
```

223.3 Parameters

- **rgb_string** (String): A comma-separated string of color values (e.g., “255,0,0” for red or “0,0,255,128” for semi-transparent blue).

223.4 Return Values

Returns Empty upon completion.

223.5 Remarks

- Color values should be between 0 and 255.
- If four values are provided, the last value is the alpha (transparency) channel.

223.6 Code Example

```
<%
Dim img
Set img = Server.CreateObject("G3IMAGE")
If img.NewContext(100, 100) Then
    ' Set to red
    img.SetColor "255,0,0"
    img.DrawCircle 50, 50, 40
    img.Fill()
End If
Set img = Nothing
%>
```

224 SetHexColor Method

224.1 Overview

Sets the active color for drawing and filling operations using a hexadecimal string in the G3Pix AxonASP G3IMAGE library.

224.2 Syntax

```
obj.SetHexColor(hex_string)
```

224.3 Parameters

- **hex_string** (String): A hexadecimal color string (e.g., “#FF0000” or “00FF00”). The “#” prefix is optional.

224.4 Return Values

Returns Empty upon completion.

224.5 Remarks

- Supports 3, 4, 6, and 8 digit hex codes.

- 4 and 8 digit codes include an alpha (transparency) channel.

224.6 Code Example

```
<%
Dim img
Set img = Server.CreateObject("G3IMAGE")
If img.NewContext(100, 100) Then
    ' Set to blue
    img.SetHexColor "#0000FF"
    img.DrawRectangle 10, 10, 80, 80
    img.Stroke()
End If
Set img = Nothing
%>
```

225 SetLineWidth Method

225.1 Overview

Sets the width of the line for subsequent stroke operations in the G3Pix AxonASP G3IMAGE library.

225.2 Syntax

```
obj.SetLineWidth(width)
```

225.3 Parameters

- **width** (Double): The thickness of the line in pixels.

225.4 Return Values

Returns Empty upon completion.

225.5 Remarks

- This setting affects all future Stroke calls until it is changed again.

225.6 Code Example

```
<%
Dim img
Set img = Server.CreateObject("G3IMAGE")
If img.NewContext(100, 100) Then
    img.SetLineWidth 5.5
    img.DrawLine 0, 0, 100, 100
    img.Stroke()
End If
Set img = Nothing
%>
```

226 Stroke Method

226.1 Overview

Outlines the current path using the active color and line width in the G3Pix AxonASP G3IMAGE library.

226.2 Syntax

```
obj.Stroke()
```

226.3 Return Values

Returns Empty upon completion.

226.4 Remarks

- Use SetColor or SetHexColor and SetLineWidth to define the appearance before calling this method.
- The path is cleared after the stroke operation. Use StrokePreserve if you need to keep the path.

226.5 Code Example

```
<%
Dim img
Set img = Server.CreateObject("G3IMAGE")
If img.NewContext(200, 200) Then
    img.SetHexColor("#000000")
    img.SetLineWidth 2
    img.DrawCircle 100, 100, 80
    img.Stroke()
End If
Set img = Nothing
%>
```

227 DefaultFormat Property

227.1 Overview

Gets or sets the default image format used for rendering operations in the G3Pix AxonASP G3IMAGE library.

227.2 Syntax

```
' Get the current format
format = obj.DefaultFormat

' Set a new format
obj.DefaultFormat = "jpg"
```

227.3 Return Values

Returns a String containing the current default format (e.g., “png” or “jpg”).

227.4 Remarks

- The default value is “png”.
- Supported values are “png”, “jpg”, and “jpeg”.

227.5 Code Example

```
<%
Dim img
Set img = Server.CreateObject("G3IMAGE")
img.DefaultFormat = "jpg"
Response.Write "Default format: " & img.DefaultFormat
Set img = Nothing
%>
```

228 HasContext Property

228.1 Overview

Indicates whether a drawing context has been initialized for the G3Pix AxonASP G3IMAGE library.

228.2 Syntax

```
exists = obj.HasContext
```

228.3 Return Values

Returns a Boolean value. True if a context is active; otherwise, False.

228.4 Remarks

- A context is typically created using the NewContext method.
- This property is read-only.

228.5 Code Example

```
<%
Dim img
Set img = Server.CreateObject("G3IMAGE")
If Not img.HasContext Then
    img.NewContext 400, 300
End If
Set img = Nothing
%>
```

229 Height Property

229.1 Overview

Returns the height of the current drawing context in the G3Pix AxonASP G3IMAGE library.

229.2 Syntax

```
h = obj.Height
```

229.3 Return Values

Returns an Integer representing the height in pixels. Returns 0 if no context is active.

229.4 Remarks

- This property is read-only. To set the height, create a new context using NewContext.

229.5 Code Example

```
<%
Dim img
Set img = Server.CreateObject("G3IMAGE")
img.NewContext 400, 300
Response.Write "Context height: " & img.Height
Set img = Nothing
%>
```

230 JpgQuality Property

230.1 Overview

Gets or sets the quality level for JPEG encoding in the G3Pix AxonASP G3IMAGE library.

230.2 Syntax

```
' Get current quality
q = obj.JpgQuality

' Set new quality
obj.JpgQuality = 85
```

230.3 Return Values

Returns an Integer representing the quality level (1 to 100).

230.4 Remarks

- The default quality is 90.

- Higher values result in better image quality but larger file sizes.

230.5 Code Example

```
<%
Dim img
Set img = Server.CreateObject("G3IMAGE")
img.JpgQuality = 70
' Perform save or render operations
Set img = Nothing
%>
```

231 LastBytes Property

231.1 Overview

Returns the binary data of the last rendered or loaded image in the G3Pix AxonASP G3IMAGE library.

231.2 Syntax

```
data = obj.LastBytes
```

231.3 Return Values

Returns a VBArray containing binary bytes.

231.4 Remarks

- This property is populated after calling methods like RenderViaTemp.
- It is read-only.

231.5 Code Example

```
<%
Dim img, bytes
Set img = Server.CreateObject("G3IMAGE")
img.NewContext 100, 100
img.RenderViaTemp "png", 0
bytes = img.LastBytes
' bytes now contains the binary PNG data
Set img = Nothing
%>
```

232 LastError Property

232.1 Overview

Returns the error message for the last operation performed in the G3Pix AxonASP G3IMAGE library.

232.2 Syntax

```
err = obj.LastError
```

232.3 Return Values

Returns a String containing the last error message. Returns an empty string if no error occurred.

232.4 Remarks

- Check this property after any operation that returns False or Empty to understand why it failed.
- This property is read-only.

232.5 Code Example

```
<%
Dim img
Set img = Server.CreateObject("G3IMAGE")
If Not img.LoadImage("missing.png") Then
    Response.Write "Error: " & img.LastError
End If
Set img = Nothing
%>
```

233 LastMimeType Property

233.1 Overview

Returns the MIME type of the last image rendered or loaded in the G3Pix AxonASP G3IMAGE library.

233.2 Syntax

```
mime = obj.LastMimeType
```

233.3 Return Values

Returns a String containing the MIME type (e.g., “image/png” or “image/jpeg”).

233.4 Remarks

- This property can be used to set the `Response.ContentType` before streaming binary content.
- It is read-only.

233.5 Code Example

```
<%
Dim img
```

```
Set img = Server.CreateObject("G3IMAGE")
img.NewContext 100, 100
img.RenderViaTemp "png", 0
Response.ContentType = img.LastMimeType
Response.BinaryWrite img.LastBytes
Set img = Nothing
%>
```

234 LastTempFile Property

234.1 Overview

Returns the file path of the temporary file used during the last rendering operation in the G3Pix AxonASP G3IMAGE library.

234.2 Syntax

```
path = obj.LastTempFile
```

234.3 Return Values

Returns a String containing the full file path.

234.4 Remarks

- The file is typically deleted automatically after the operation completes.
- This property is primarily used for debugging purposes.
- It is read-only.

234.5 Code Example

```
<%
Dim img
Set img = Server.CreateObject("G3IMAGE")
img.NewContext 100, 100
img.RenderViaTemp "png", 0
Response.Write "Temp file used: " & img.LastTempFile
Set img = Nothing
%>
```

235 Width Property

235.1 Overview

Returns the width of the current drawing context in the G3Pix AxonASP G3IMAGE library.

235.2 Syntax

```
w = obj.Width
```

235.3 Return Values

Returns an Integer representing the width in pixels. Returns 0 if no context is active.

235.4 Remarks

- This property is read-only. To set the width, create a new context using NewContext.

235.5 Code Example

```
<%
Dim img
Set img = Server.CreateObject("G3IMAGE")
img.NewContext 400, 300
Response.Write "Context width: " & img.Width
Set img = Nothing
%>
```

236 Use the G3FILES Library

236.1 Overview

The **G3FILES** library provides high-performance file system operations for G3Pix AxonASP applications. It offers a comprehensive set of methods for reading, writing, and managing files and directories with built-in support for multiple text encodings (UTF-8, UTF-16, ISO-8859-1, ASCII) and line ending normalization. The library is optimized for server-side file management and operates within the AxonASP sandboxed file system.

236.2 Syntax

To instantiate the library, use the following syntax:

```
Set files = Server.CreateObject("G3FILES")
```

236.3 Prerequisites

- **File System Permissions**: The AxonASP service must have appropriate read and write permissions for the target directories.
- **Sandbox Configuration**: All file paths are resolved relative to the configured AxonASP sandbox root for security.

236.4 How it Works

The G3FILES object provides direct access to the underlying host file system through the AxonASP Virtual Machine. - **Encoding Support**: The library automatically handles Byte Order Marks (BOM) when reading files and allows explicit BOM inclusion when writing. - **Line Endings**: Line endings can be normalized to Windows (CRLF),

Linux/Unix (LF), or legacy Mac (CR) styles during write or conversion operations. - **Path Resolution**: Relative paths are automatically resolved within the AxonASP sandbox, ensuring that file operations cannot escape the intended application directory.

236.5 API Reference

236.5.1 Methods

- **Append**: Appends text content to an existing file or creates it if it does not exist.
- **ConvertFileEncoding**: Converts the text encoding and line endings of a file on disk.
- **ConvertTextEncoding**: Returns a string converted from one text encoding to another.
- **Copy**: Copies a file from a source path to a destination path.
- **Delete**: Permanently removes a file or directory from the file system.
- **Exists**: Returns a Boolean indicating whether a file or directory exists at the specified path.
- **List**: Returns an Array containing the names of all files within a specified directory.
- **Mkdir**: Creates a new directory or a full directory path recursively.
- **Move**: Renames or moves a file or directory to a new location.
- **NormalizeEOL**: Returns a string with line endings converted to a specified style.
- **Read**: Returns the full text content of a file using a specified encoding.
- **Size**: Returns the size of a file in bytes.
- **Write**: Writes text content to a file, overwriting any existing content.

236.6 Code Example

The following example demonstrates how to check for a file's existence, read its content, and write a modified version to a new location.

```
<%
Dim files, content, success
Set files = Server.CreateObject("G3FILES")

' Check if a configuration file exists
If files.Exists("/config/settings.txt") Then
    ' Read content using UTF-8 encoding
    content = files.Read("/config/settings.txt", "utf-8")

    ' Append a timestamp to the content
    content = content & vbCrLf & "Last Access: " & Now()
```

```
    ' Write the updated content to a backup file
    success = files.Write("/backups/settings_backup.txt", content, "utf-8
", "windows", True)

    If success Then
        Response.Write "Backup created successfully."
    Else
        Response.Write "Failed to create backup."
    End If
Else
    Response.Write "Source file not found."
End If

Set files = Nothing
%>
```

237 G3FILES Methods

237.1 Overview

This page summarizes every method exposed by `G3FILES` in G3Pix AxonASP.

237.2 Methods

Method	Returns	Description
`Exists(path)`	Boolean	Returns `True` when the file or directory exists; otherwise `False`.
`Read(path [, encoding])`	String	Returns file text decoded with the requested encoding; returns empty string when path is missing or read fails.
`Write(path, content [, encoding] [, lineEnding] [, includeBOM])`	Boolean	Writes content to file (overwrite mode). Returns `True` on success; otherwise `False`.
`Append(path, content [, encoding] [, lineEnding])`	Boolean	Appends content to file (create if missing). Returns `True` on success; otherwise `False`.
`Delete(path)`	Boolean	Removes a file or directory entry. Returns `True` on

Method	Returns	Description
		success; otherwise `False`.
`Copy(sourcePath, destPath)`	Boolean	Copies source file to destination path. Returns `True` on success; otherwise `False`.
`Move(sourcePath, destPath)`	Boolean	Moves or renames source path to destination path. Returns `True` on success; otherwise `False`.
`Size(path)`	Integer	Returns file size in bytes, or `0` when path is missing, file does not exist, or path is a directory.
`Mkdir(path)`	Boolean	Creates directory path recursively. Returns `True` on success; otherwise `False`.
`List(path)`	Array	Returns a VBScript array of file names in the directory; returns an empty array when path is missing or directory read fails.
`NormalizeEOL(text, style)`	String	Returns text with normalized line endings (`windows`, `mac`, or `linux` style aliases).
`ConvertTextEncoding(text, sourceEncoding, destEncoding)`	String	Returns converted text; returns empty string when required arguments are missing.
`ConvertFileEncoding(sourcePath, destPath, sourceEncoding, destEncoding [, lineEnding] [, includeBOM])`	Boolean	Converts file encoding and optionally line endings/BOM. Returns `True` on success; otherwise `False`.

237.3 Remarks

- Instantiate the library with `Server.CreateObject("G3FILES")`.

- Method names are case-insensitive.
- Operations are sandbox-path aware and resolve through the runtime host.

238 G3FILES Properties

238.1 Overview

This page lists properties exposed by `G3FILES`.

238.2 Properties

Property	Access	Type	Description
None	N/A	N/A	`G3FILES` does not expose public properties. Use methods for all operations.

238.3 Remarks

- Instantiate the library with `Server.CreateObject("G3FILES")`.

239 Append Method

239.1 Overview

Appends text content to an existing file or creates a new file if it does not already exist.

239.2 Syntax

```
success = files.Append(path, content [, encoding] [, lineEnding])
```

239.3 Parameters and Arguments

- **path** (String, Required): The target file path relative to the AxonASP sandbox.
- **content** (String, Required): The text to append to the file.
- **encoding** (String, Optional): The text encoding to use (e.g., “utf-8”, “utf-16”, “ascii”, “iso-8859-1”). The default is “utf-8”.
- **lineEnding** (String, Optional): The line ending style to apply (e.g., “windows”, “linux”, “mac”). The default is “linux” (LF).

239.4 Return Values

Returns a **Boolean** indicating whether the append operation was successful.

239.5 Remarks

- If the file exists, the new content is added to the end.
- If the file does not exist, it is created.
- The library automatically handles directory creation if the target path includes missing folders.
- This method is also accessible via the **AppendText** alias.

239.6 Code Example

```
<%
Dim files, logPath
Set files = Server.CreateObject("G3FILES")
logPath = "/logs/app_log.txt"
If files.Append(logPath, Now() & " - User logged in", "utf-8", "windows")
 Then
    Response.Write "Log entry added."
End If
Set files = Nothing
%>
```

240 ConvertFileEncoding Method

240.1 Overview

Converts the text encoding and line endings of a file on disk and saves the result to a destination path.

240.2 Syntax

```
success = files.ConvertFileEncoding(source, dest, srcEnc, dstEnc [, lineE
nding] [, includeBOM])
```

240.3 Parameters and Arguments

- **source** (String, Required): The path to the source file.
- **dest** (String, Required): The path to the destination file.
- **srcEnc** (String, Required): The encoding of the source file.
- **dstEnc** (String, Required): The target encoding for the destination file.
- **lineEnding** (String, Optional): The target line ending style (e.g., “windows”, “linux”).
- **includeBOM** (Boolean, Optional): Specifies whether to include a Byte Order Mark in the destination file.

240.4 Return Values

Returns a **Boolean** indicating whether the conversion and save operation were successful.

240.5 Remarks

- This method is useful for batch-processing files for cross-platform compatibility.
- Supported encodings include “utf-8”, “utf-16”, “ascii”, and “iso-8859-1”.

240.6 Code Example

```
<%
Dim files
Set files = Server.CreateObject("G3FILES")
' Convert an ISO-8859-1 file to UTF-8 with Windows line endings and a BOM
If files.ConvertFileEncoding("legacy.txt", "modern.txt", "iso-8859-1", "u
tf-8", "windows", True) Then
    Response.Write "File converted successfully."
End If
Set files = Nothing
%>
```

241 ConvertTextEncoding Method

241.1 Overview

Returns a string that has been converted from one text encoding to another.

241.2 Syntax

```
convertedText = files.ConvertTextEncoding(text, fromEnc, toEnc)
```

241.3 Parameters and Arguments

- **text** (String, Required): The source text to be converted.
- **fromEnc** (String, Required): The current encoding of the text.
- **toEnc** (String, Required): The target encoding for the result.

241.4 Return Values

Returns a **String** containing the converted text.

241.5 Remarks

- This method is performed in memory and does not affect any files on disk.
- It is useful for processing data received from external sources with different encoding requirements.

241.6 Code Example

```
<%
Dim files, sourceText, targetText
Set files = Server.CreateObject("G3FILES")
sourceText = "Special characters: é à ö"
```

```
targetText = files.ConvertTextEncoding(sourceText, "iso-8859-1", "utf-8")
Response.Write "Converted text: " & targetText
Set files = Nothing
%>
```

242 Copy Method

242.1 Overview

Copies a file from a source path to a destination path.

242.2 Syntax

```
success = files.Copy(source, dest)
```

242.3 Parameters and Arguments

- **source** (String, Required): The path to the file to be copied.
- **dest** (String, Required): The target path for the copy.

242.4 Return Values

Returns a **Boolean** indicating whether the copy operation was successful.

242.5 Remarks

- If the destination file already exists, it will be overwritten.
- Both paths are relative to the AxonASP sandbox root.

242.6 Code Example

```
<%
Dim files
Set files = Server.CreateObject("G3FILES")
If files.Copy("/data/source.txt", "/backups/copy.txt") Then
    Response.Write "File copied."
End If
Set files = Nothing
%>
```

243 Delete Method

243.1 Overview

Permanently removes a file or directory from the file system.

243.2 Syntax

```
success = files.Delete(path)
```

243.3 Parameters and Arguments

- **path** (String, Required): The target file or directory path to be removed.

243.4 Return Values

Returns a **Boolean** indicating whether the deletion was successful.

243.5 Remarks

- If the target path is a directory, it must be empty before it can be deleted.
- This method is also accessible via the **Remove** alias.
- Path resolution is relative to the AxonASP sandbox root.

243.6 Code Example

```
<%
Dim files
Set files = Server.CreateObject("G3FILES")
If files.Delete("/temp/old_data.txt") Then
    Response.Write "File deleted."
Else
    Response.Write "Failed to delete file."
End If
Set files = Nothing
%>
```

244 Exists Method

244.1 Overview

Returns a Boolean indicating whether a file or directory exists at the specified path.

244.2 Syntax

```
boolExists = files.Exists(path)
```

244.3 Parameters and Arguments

- **path** (String, Required): The target path to check.

244.4 Return Values

Returns a **Boolean** value. It returns **True** if the file or directory exists, and **False** otherwise.

244.5 Remarks

- Path resolution is relative to the AxonASP sandbox root.
- This method is efficient for validating resources before attempting read or write operations.

244.6 Code Example

```
<%
Dim files
Set files = Server.CreateObject("G3FILES")
If files.Exists("/web.config") Then
    Response.Write "Configuration file exists."
Else
    Response.Write "File not found."
End If
Set files = Nothing
%>
```

245 List Method

245.1 Overview

Returns an Array containing the names of all files within a specified directory.

245.2 Syntax

```
fileNamesArray = files.List(path)
```

245.3 Parameters and Arguments

- **path** (String, Required): The target directory path to list.

245.4 Return Values

Returns an **Array** (Variant) containing the names of the files. If the directory is empty or does not exist, it returns an empty array.

245.5 Remarks

- The returned array only includes file names, not subdirectories.
- Path resolution is relative to the AxonASP sandbox root.
- This method is also accessible via the **ListFiles** alias.

245.6 Code Example

```
<%
Dim files, fileList, i
Set files = Server.CreateObject("G3FILES")
fileList = files.List("/uploads")

If UBound(fileList) >= 0 Then
    Response.Write "Found " & UBound(fileList) + 1 & " files:<br>"
    For i = 0 To UBound(fileList)
        Response.Write "- " & fileList(i) & "<br>"
    Next
Else
```

```
    Response.Write "No files found in directory."
End If
Set files = Nothing
%>
```

246 Mkdir Method

246.1 Overview

Creates a new directory or a full directory path recursively.

246.2 Syntax

```
success = files.Mkdir(path)
```

246.3 Parameters and Arguments

- **path** (String, Required): The directory path to be created.

246.4 Return Values

Returns a **Boolean** indicating whether the directory creation was successful.

246.5 Remarks

- This method is recursive; it will create any missing parent directories in the path automatically.
- If the directory already exists, the method returns **True**.
- This method is also accessible via the **MakeDir** alias.

246.6 Code Example

```
<%
Dim files
Set files = Server.CreateObject("G3FILES")
' Creates the entire directory structure recursively
If files.Mkdir("/data/logs/archives/2026") Then
    Response.Write "Directory path created."
End If
Set files = Nothing
%>
```

247 Move Method

247.1 Overview

Renames or moves a file or directory to a new location in the file system.

247.2 Syntax

```
success = files.Move(source, dest)
```

247.3 Parameters and Arguments

- **source** (String, Required): The path to the file or directory to be moved.
- **dest** (String, Required): The target path for the move or rename.

247.4 Return Values

Returns a **Boolean** indicating whether the move or rename operation was successful.

247.5 Remarks

- If the destination already exists, the move operation will fail.
- This method is also accessible via the **Rename** alias.
- Path resolution is relative to the AxonASP sandbox root.

247.6 Code Example

```
<%
Dim files
Set files = Server.CreateObject("G3FILES")
If files.Move("/temp/data.csv", "/data/processed/data.csv") Then
    Response.Write "File moved."
End If
Set files = Nothing
%>
```

248 NormalizeEOL Method

248.1 Overview

Returns a string with line endings converted to a specified style.

248.2 Syntax

```
normalizedText = files.NormalizeEOL(text, style)
```

248.3 Parameters and Arguments

- **text** (String, Required): The source text containing inconsistent line endings.
- **style** (String, Required): The target line ending style (e.g., “windows”, “linux”, “mac”).

248.4 Return Values

Returns a **String** with the normalized line endings.

248.5 Remarks

- Styles include “windows” (CRLF), “linux” (LF), and “mac” (CR).
- This method is performed in memory and does not affect any files on disk.

- This method is also accessible via the **NormalizeLineEndings** alias.

248.6 Code Example

```
<%
Dim files, rawText, normalizedText
Set files = Server.CreateObject("G3FILES")
rawText = "Line 1" & Chr(10) & "Line 2" & Chr(13) & "Line 3"
normalizedText = files.NormalizeEOL(rawText, "windows")
' All line endings are now CRLF
Set files = Nothing
%>
```

249 Read Method

249.1 Overview

Returns the full text content of a file using a specified encoding.

249.2 Syntax

```
fileContent = files.Read(path [, encoding])
```

249.3 Parameters and Arguments

- **path** (String, Required): The path to the file to be read.
- **encoding** (String, Optional): The text encoding to use for decoding the file (e.g., “utf-8”, “utf-16”, “ascii”, “iso-8859-1”). The default is “utf-8”.

249.4 Return Values

Returns a **String** containing the full content of the file. If the file cannot be read, it returns an empty string.

249.5 Remarks

- The library automatically detects Byte Order Marks (BOM) to determine the correct encoding.
- This method is also accessible via the **ReadText** alias.
- Path resolution is relative to the AxonASP sandbox root.

249.6 Code Example

```
<%
Dim files, content
Set files = Server.CreateObject("G3FILES")
content = files.Read("/config/app.json", "utf-8")
If content <> "" Then
    Response.Write "File read successfully."
End If
```

```
Set files = Nothing
%>
```

250 Size Method

250.1 Overview

Returns the size of a specified file in bytes.

250.2 Syntax

```
fileSize = files.Size(path)
```

250.3 Parameters and Arguments

- **path** (String, Required): The path to the file.

250.4 Return Values

Returns an **Integer** (int64) representing the size of the file in bytes. If the file does not exist or is a directory, it returns 0.

250.5 Remarks

- This method provides a quick way to check file metadata without reading the content.
- Path resolution is relative to the AxonASP sandbox root.

250.6 Code Example

```
<%
Dim files, s
Set files = Server.CreateObject("G3FILES")
s = files.Size("/backups/dump.sql")
Response.Write "Backup size: " & FormatNumber(s / 1024, 2) & " KB"
Set files = Nothing
%>
```

251 Write Method

251.1 Overview

Writes text content to a file, overwriting any existing content.

251.2 Syntax

```
success = files.Write(path, content [, encoding] [, lineEnding] [, includ
eBOM])
```

251.3 Parameters and Arguments

- **path** (String, Required): The target file path.

- **content** (String, Required): The text content to write.
- **encoding** (String, Optional): The text encoding to use (e.g., “utf-8”, “utf-16”, “ascii”, “iso-8859-1”). The default is “utf-8”.
- **lineEnding** (String, Optional): The line ending style to apply (e.g., “windows”, “linux”). The default is “linux” (LF).
- **includeBOM** (Boolean, Optional): Specifies whether to include a Byte Order Mark in the file. The default is **False**.

251.4 Return Values

Returns a **Boolean** indicating whether the write operation was successful.

251.5 Remarks

- If the file already exists, it is overwritten.
- The library automatically handles directory creation if the target path includes missing folders.
- This method is also accessible via the **WriteText** alias.

251.6 Code Example

```
<%
Dim files, content, success
Set files = Server.CreateObject("G3FILES")
content = "Configuration Data" & vbCrLf & "Version: 2.1"
' Write as UTF-8 with a BOM and Windows line endings
success = files.Write("config.txt", content, "utf-8", "windows", True)
If success Then
    Response.Write "Configuration saved."
End If
Set files = Nothing
%>
```

252 Use the G3TESTSUITE Library

252.1 Overview

The **G3TESTSUITE** library provides a native assertion framework for G3Pix AxonASP. It is the primary tool for writing automated unit tests and integration tests within the AxonASP environment. The library enables developers to verify variable states, compare values using native VBScript coercion, and validate error-handling logic.

252.2 Syntax

To instantiate the test suite, use the following syntax:

```
Set t = Server.CreateObject("G3TESTSUITE")
```

252.3 Prerequisites

No external dependencies are required. The G3TESTSUITE library is a built-in component of the AxonASP Virtual Machine and is designed to work seamlessly with the `axonasp-testsuite` runner.

252.4 How it Works

The G3TESTSUITE object maintains internal counters for total, passed, and failed assertions. When an assertion method (such as **AssertEqual**) is called, the library evaluates the condition using the same comparison logic as the AxonASP VM, ensuring that test behavior perfectly matches production behavior.

If an assertion fails, the library records a failure message, including the active description set by **Describe**. These failures are aggregated by the AxonASP test runner to produce structured reports.

252.5 API Reference

252.5.1 Methods

- **AssertEmpty**: Verifies that a value is the VBScript `Empty` variant.
- **AssertEqual**: Verifies that two values are equal using standard coercion.
- **AssertFalse**: Verifies that a condition evaluates to `False`.
- **AssertLength**: Verifies the number of elements in an array or the length of a string.
- **AssertNothing**: Verifies that a value is the VBScript `Nothing` object.
- **AssertNotEqual**: Verifies that two values are not equal.
- **AssertNull**: Verifies that a value is the VBScript `Null` variant.
- **AssertRaises**: Verifies that a VBScript code block raises a specific error.
- **AssertTrue**: Verifies that a condition evaluates to `True`.
- **AssertTypeName**: Verifies the VBScript `TypeName` of a value.
- **Describe**: Sets a label for the current block of assertions.
- **Fail**: Explicitly records an assertion failure.

252.5.2 Properties

- **Failed**: Returns the number of failed assertions in the current instance.
- **HasFailures**: Returns a Boolean indicating if any assertions have failed.
- **Passed**: Returns the number of passed assertions in the current instance.
- **Suite**: Gets or sets the current description label.
- **Total**: Returns the total number of assertions executed.

252.6 Code Example

The following example demonstrates how to perform a series of assertions to validate a business logic function.

```
<%
Dim t, result
Set t = Server.CreateObject("G3TESTSUITE")

t.Describe "Financial Calculator - Interest Rates"

' Validate simple calculation
result = CalculateInterest(1000, 0.05, 1)
t.AssertEqual 50, result, "Interest on 1000 at 5% for 1 year should be 50
"


' Validate type safety
t.AssertTypeName "Double", result, "Result must be a Double type"

' Validate error handling for negative values
t.AssertRaises "CalculateInterest -100, 0.05, 1", 5, "Should raise Illega
l Function Call"

Response.Write "Tests completed. Passed: " & t.Passed & "/" & t.Total
Set t = Nothing
%>
```

253 G3TESTSUITE Methods

253.1 Overview

This page summarizes every method exposed by `G3TESTSUITE` in G3Pix AxonASP.

253.2 Methods

Method	Returns	Description
`AssertEqual(expected, actual [, message])`	Boolean	Returns `True` when values are equal under VM comparison semantics; otherwise `False`.
`AssertNotEqual(expected, actual [, message])`	Boolean	Returns `True` when values are not equal; otherwise `False`.
`AssertTrue(condition [, message])`	Boolean	Returns `True` when condition evaluates to true; otherwise `False`.

Method	Returns	Description
`AssertFalse(condition [, message])`	Boolean	Returns `True` when condition evaluates to false; otherwise `False`.
`AssertEmpty(value [, message])`	Boolean	Returns `True` when value is Empty; otherwise `False`.
`AssertNull(value [, message])`	Boolean	Returns `True` when value is Null; otherwise `False`.
`AssertNothing(value [, message])`	Boolean	Returns `True` when value is Nothing; otherwise `False`.
`AssertTypeName(value, expectedTypeName [, message])`	Boolean	Returns `True` when `TypeName(value)` matches expected type name; otherwise `False`.
`AssertLength(value, expectedLength [, message])`	Boolean	Returns `True` when length/count matches expected value; otherwise `False`.
`AssertRaises(code [, expected] [, message])`	Boolean	Executes code and returns `True` when an expected error condition is met; otherwise `False`.
`Fail([message])`	Boolean	Records an explicit failure and returns `False`.
`Describe(label)`	Empty	Sets current suite/describe label used in failure diagnostics.

253.3 Remarks

- Instantiate the library with `Server.CreateObject("G3TESTSUITE")`.
- Method names are case-insensitive.
- Assertion methods update `Total`, `Passed`, and `Failed` counters.

254 G3TESTSUITE Properties

254.1 Overview

This page lists the properties exposed by `G3TESTSUITE`.

254.2 Properties

Property	Access	Type	Description
`Suite`	Read/Write	String	Current suite/describe label for assertion diagnostics.
`CurrentDescribe`	Read/Write	String	Alias of `Suite` for describe label access.
`Total`	Read-only	Integer	Total number of assertions executed.
`Passed`	Read-only	Integer	Number of successful assertions.
`Failed`	Read-only	Integer	Number of failed assertions.
`HasFailures`	Read-only	Boolean	`True` when one or more failures are recorded.

254.3 Remarks

- Instantiate the library with `Server.CreateObject("G3TESTSUITE")`.
- `Suite` and `CurrentDescribe` map to the same runtime field.

255 Use the G3TEMPLATE Library

255.1 Overview

The **G3TEMPLATE** library provides a high-performance template rendering engine for G3Pix AxonASP. It allows developers to separate application logic from presentation by using external template files. The library leverages the powerful Go `html/template` engine, providing secure, context-aware HTML escaping and data binding.

255.2 Syntax

To instantiate the library, use the following syntax:

```
Set template = Server.CreateObject("G3TEMPLATE")
```

255.3 Prerequisites

- **Template Files**: Ensure your template files (typically `.html` or `.tmpl`) are accessible within the server's file system.
- **Data Structures**: Data passed to templates should ideally be in the form of a **Scripting.Dictionary** or a standard **Array**.

255.4 How it Works

The G3TEMPLATE object parses an external file and executes it using a provided data object. - **Data Binding**: You can pass complex nested structures (Dictionaries within Arrays, or vice-versa) to the template. - **Path Resolution**: The library automatically resolves relative paths using `Server.MapPath`. - **Security**: The underlying engine provides automatic contextual escaping, protecting your application against Cross-Site Scripting (XSS) attacks by default.

255.5 API Reference

255.5.1 Methods

- **Render**: Parses a template file and returns a string containing the rendered output.

255.6 Code Example

The following example demonstrates how to pass a dictionary of data to a template and render the result.

```
<%
Dim template, data, html
Set template = Server.CreateObject("G3TEMPLATE")

' Prepare data for the template
Set data = Server.CreateObject("Scripting.Dictionary")
data.Add "Title", "Welcome to AxonASP"
data.Add "User", "Lucas"

' Render the template located at /views/index.html
html = template.Render("/views/index.html", data)

' Output the rendered HTML
Response.Write html

Set template = Nothing
%>
```

256 G3TEMPLATE Methods

256.1 Overview

This page summarizes every method exposed by `G3TEMPLATE` in G3Pix AxonASP.

256.2 Methods

Method	Returns	Description
`Render(templatePath [, data])`	String	Renders a template file and returns output text; returns an error String when path is missing, template parsing fails, or template execution fails.

256.3 Remarks

- Instantiate the library with `Server.CreateObject("G3TEMPLATE")`.
- Method names are case-insensitive.
- Template paths are resolved through the host environment when available.

257 G3TEMPLATE Properties

257.1 Overview

This page lists properties exposed by `G3TEMPLATE`.

257.2 Properties

Property	Access	Type	Description
None	N/A	N/A	`G3TEMPLATE` does not expose public properties. Use methods for all operations.

257.3 Remarks

- Instantiate the library with `Server.CreateObject("G3TEMPLATE")`.

258 Render Method

258.1 Overview

Parses an external template file and returns a string containing the rendered output using an optional data object in the G3Pix AxonASP G3TEMPLATE library.

258.2 Syntax

```
renderedOutput = obj.Render(templatePath [, data])
```

258.3 Parameters and Arguments

- **templatePath** (String, Required): The relative or absolute path to the template file on the server.
- **data** (Variant, Optional): An object or array containing the data to be bound to the template. This is typically a **Scripting.Dictionary** or a standard **Array**.

258.4 Return Values

Returns a **String** containing the fully rendered template output. If an error occurs during parsing or execution (e.g., file not found or invalid template syntax), the method returns a **String** prefixed with "Error:" followed by the error description.

258.5 Remarks

- The template engine uses standard Go `html/template` syntax (e.g., `{{ .ValueName }}`).
- The method automatically resolves relative paths using `Server.MapPath`.
- Context-aware escaping is automatically applied to prevent XSS.

258.6 Code Example

```
<%
Dim template, user, output
Set template = Server.CreateObject("G3TEMPLATE")

' Define a data dictionary
Set user = Server.CreateObject("Scripting.Dictionary")
user.Add "Name", "Lucas"
user.Add "IsAdmin", True

' Render the template
output = template.Render("/templates/profile.html", user)

' Check for errors and display result
If Left(output, 6) = "Error:" Then
    Response.Write "Rendering Failed: " & output
Else
    Response.Write output
End If

Set template = Nothing
%>
```

259 Use the G3ZIP Library

259.1 Overview

The **G3ZIP** library provides high-performance compression and archive management services for G3Pix AxonASP applications. It enables developers to create, read, and manipulate ZIP archives directly on the server without external dependencies. The library supports adding files and folders, generating archives from dynamic text, and extracting content to the file system.

259.2 Syntax

To instantiate the library, use the following syntax:

```
Set zip = Server.CreateObject("G3ZIP")
```

259.3 Prerequisites

- **File System Permissions**: The AxonASP process must have read/write access to the paths where archives are created or extracted.
- **Path Resolution**: Relative paths are automatically resolved via `Server.MapPath` in a web context.

259.4 How it Works

The G3ZIP object operates in two primary modes: **Read** and **Write**. - **Read Mode**: Initialized using the **Open** method. In this mode, you can inspect the archive content, retrieve file metadata, and extract files. - **Write Mode**: Initialized using the **Create** method. In this mode, you can build a new archive by adding existing files, entire directory structures, or custom text strings as virtual files.

The library ensures proper resource management by requiring a call to the **Close** method when operations are complete, which flushes buffers and releases file handles.

259.5 API Reference

259.5.1 Methods

- **AddFile**: Includes an existing file into the active archive.
- **AddFolder**: Recursively includes a directory and its contents into the active archive.
- **AddText**: Creates a virtual file in the archive from a string.
- **Close**: Finalizes the archive and releases system resources.
- **Create**: Initializes a new ZIP archive for writing.
- **ExtractAll**: Unpacks the entire archive to a specified directory.

- **ExtractFile**: Unpacks a single file from the archive.
- **GetInfo**: Retrieves technical metadata for a specific file within the archive.
- **List**: Returns an array of file names contained in the archive.
- **Open**: Opens an existing ZIP archive for reading.

259.5.2 Properties

- **Count**: Returns the number of files currently in the archive (Read mode).
- **Mode**: Indicates if the object is in Read ("r") or Write ("w") mode.
- **Path**: Returns the full physical path of the active archive.

259.6 Code Example

The following example demonstrates how to create a ZIP archive containing a physical file and a dynamic text file.

```
<%
Dim zip, success
Set zip = Server.CreateObject("G3ZIP")

' Initialize a new archive
If zip.Create("/downloads/package.zip") Then
    ' Add a physical file from the server
    zip.AddFile "/data/report.pdf", "report.pdf"

    ' Add dynamic content as a text file
    zip.AddText "readme.txt", "This archive was generated by AxonASP on "
 & Now()

    ' Finalize the archive
    zip.Close
    Response.Write "Archive created successfully."
Else
    Response.Write "Failed to create archive."
End If

Set zip = Nothing
%>
```

260 G3ZIP Methods

260.1 Overview

This page summarizes every method exposed by `G3ZIP` in G3Pix AxonASP.

260.2 Methods

Method	Returns	Description

Method	Returns	Description
`Open(path)`	Boolean	Opens an existing ZIP archive for read mode. Returns `True` on success; otherwise `False`.
`Create(path)`	Boolean	Creates a ZIP archive for write mode. Returns `True` on success; otherwise `False`.
`AddFile(sourcePath [, nameInZip])`	Boolean	Adds one file to an archive in write mode. Returns `True` on success; otherwise `False`.
`AddFolder(sourcePath [, nameInZip])`	Boolean	Adds one folder recursively to an archive in write mode. Returns `True` on success; otherwise `False`.
`AddText(nameInZip, text)`	Boolean	Adds text content as an entry in write mode. Returns `True` on success; otherwise `False`.
`ExtractAll(destinationPath)`	Boolean	Extracts all entries in read mode. Returns `True` on success; otherwise `False`.
`ExtractFile(nameInZip, destinationPath)`	Boolean	Extracts one entry in read mode. Returns `True` on success; otherwise `False`.
`List()`	Array	Returns an array of entry names from the open archive.
`GetInfo(nameInZip)`	Scripting.Dictionary or Empty	Returns metadata dictionary for one entry, or Empty when argument is missing or entry is not found.
`Close()`	Boolean	Closes archive resources and returns `True`.

260.3 Remarks

- Instantiate the library with `Server.CreateObject("G3ZIP")`.
- Method names are case-insensitive.
- Write operations require write mode; read/extract operations require read mode.

261 G3ZIP Properties

261.1 Overview

This page lists the properties exposed by `G3ZIP`.

261.2 Properties

Property	Access	Type	Description
`Path`	Read-only	String	Full path of the active ZIP archive.
`Mode`	Read-only	String	Current mode (`r` for read or `w` for write).
`Count`	Read-only	Integer	Number of entries in the open reader archive; `0` when not in read mode.

261.3 Remarks

- Instantiate the library with `Server.CreateObject("G3ZIP")`.

262 AddFile Method

262.1 Overview

Includes a physical file into the current write-mode archive in the G3Pix AxonASP G3ZIP library.

262.2 Syntax

```
success = zip.AddFile(sourcePath [, nameInZip])
```

262.3 Parameters and Arguments

- **sourcePath** (String, Required): The path to the file on the server to be added.
- **nameInZip** (String, Optional): The relative path and name the file will have inside the ZIP archive. If omitted, the library uses the base name of the source file.

262.4 Return Values

Returns a **Boolean** indicating whether the file was successfully added to the archive.

262.5 Remarks

- The object must be in **Write** mode (initialized via the **Create** method).
- Leading slashes in `nameInZip` are automatically removed.
- Path resolution for `sourcePath` is handled relative to the AxonASP sandbox.

262.6 Code Example

```
<%
Dim zip
Set zip = Server.CreateObject("G3ZIP")
If zip.Create("/temp/archive.zip") Then
    ' Add a file and rename it within the ZIP
    zip.AddFile "/images/logo.png", "assets/branding.png"
    zip.Close
End If
Set zip = Nothing
%>
```

263 AddFolder Method

263.1 Overview

Recursively includes a directory and all of its contents into the current write-mode archive in the G3Pix AxonASP G3ZIP library.

263.2 Syntax

```
success = zip.AddFolder(sourcePath [, nameInZip])
```

263.3 Parameters and Arguments

- **sourcePath** (String, Required): The path to the directory on the server to be added.
- **nameInZip** (String, Optional): The relative path the folder will have inside the ZIP archive. If omitted, the library uses the base name of the source directory.

263.4 Return Values

Returns a **Boolean** indicating whether the folder and its contents were successfully added.

263.5 Remarks

- The object must be in **Write** mode.
- This method performs a full recursive traversal of the specified directory.

- All files and subdirectories found will preserve their relative structure within the archive.

263.6 Code Example

```
<%
Dim zip
Set zip = Server.CreateObject("G3ZIP")
If zip.Create("/backups/site_backup.zip") Then
    ' Add the entire 'content' folder to the root of the ZIP
    zip.AddFolder "/www/content", ""
    zip.Close
End If
Set zip = Nothing
%>
```

264 AddText Method

264.1 Overview

Creates a new virtual file inside the current write-mode archive using a provided string in the G3Pix AxonASP G3ZIP library.

264.2 Syntax

```
success = zip.AddText(nameInZip, content)
```

264.3 Parameters and Arguments

- **nameInZip** (String, Required): The relative path and name for the new file inside the archive.
- **content** (String, Required): The text content to be written into the file.

264.4 Return Values

Returns a **Boolean** indicating whether the virtual file was successfully created and added.

264.5 Remarks

- This method is ideal for generating dynamic files such as manifest.json, readme.txt, or configuration files on the fly.
- The object must be in **Write** mode.

264.6 Code Example

```
<%
Dim zip, info
Set zip = Server.CreateObject("G3ZIP")
If zip.Create("/temp/report.zip") Then
    info = "Report generated on: " & Now() & vbCrLf & "System: AxonASP"
```

```
    zip.AddText "metadata.txt", info
    zip.Close
End If
Set zip = Nothing
%>
```

265 Close Method

265.1 Overview

Finalizes the archive operations and releases all system resources in the G3Pix AxonASP G3ZIP library.

265.2 Syntax

```
zip.Close()
```

265.3 Return Values

Returns a **Boolean** (True) upon completion.

265.4 Remarks

- For **Write** mode, this method is mandatory to ensure all buffers are flushed and the ZIP header is correctly written to disk.
- For **Read** mode, it closes the file handle and releases the archive reader.
- After calling **Close**, the **Mode** and **Path** properties are reset.

265.5 Code Example

```
<%
Dim zip
Set zip = Server.CreateObject("G3ZIP")
zip.Create("/temp/output.zip")
' ... perform operations ...
zip.Close
Set zip = Nothing
%>
```

266 Create Method

266.1 Overview

Initializes a new ZIP archive file on the server and prepares the G3Pix AxonASP G3ZIP library for writing operations.

266.2 Syntax

```
success = zip.Create(archivePath)
```

266.3 Parameters and Arguments

- **archivePath** (String, Required): The target path where the ZIP file will be created.

266.4 Return Values

Returns a **Boolean** indicating whether the file was successfully created and initialized for writing.

266.5 Remarks

- If the target directory does not exist, the library attempts to create it recursively.
- If a file already exists at the specified path, it will be overwritten.
- Calling **Create** will close any archive currently managed by the object.

266.6 Code Example

```
<%
Dim zip
Set zip = Server.CreateObject("G3ZIP")
If zip.Create("/data/exports/data.zip") Then
    Response.Write "Write-mode initialized."
    ' Add files here...
    zip.Close
End If
Set zip = Nothing
%>
```

267 ExtractAll Method

267.1 Overview

Unpacks the entire content of the active read-mode archive into a specified directory in the G3Pix AxonASP G3ZIP library.

267.2 Syntax

```
success = zip.ExtractAll(targetDirectory)
```

267.3 Parameters and Arguments

- **targetDirectory** (String, Required): The destination directory on the server where the files will be extracted.

267.4 Return Values

Returns a **Boolean** indicating whether all files were successfully extracted.

267.5 Remarks

- The object must be in **Read** mode (initialized via the **Open** method).
- The library automatically recreates any subdirectory structure found within the archive.
- Existing files in the destination directory will be overwritten if they have the same name.

267.6 Code Example

```
<%
Dim zip
Set zip = Server.CreateObject("G3ZIP")
If zip.Open("/uploads/archive.zip") Then
    If zip.ExtractAll("/temp/extracted_files") Then
        Response.Write "Extraction complete."
    End If
    zip.Close
End If
Set zip = Nothing
%>
```

268 ExtractFile Method

268.1 Overview

Unpacks a specific file from the active read-mode archive into a specified directory in the G3Pix AxonASP G3ZIP library.

268.2 Syntax

```
success = zip.ExtractFile(fileName, targetDirectory)
```

268.3 Parameters and Arguments

- **fileName** (String, Required): The name (or relative path) of the file inside the ZIP archive to be extracted.
- **targetDirectory** (String, Required): The destination directory on the server.

268.4 Return Values

Returns a **Boolean** indicating whether the specific file was found and successfully extracted.

268.5 Remarks

- The object must be in **Read** mode.
- The `fileName` parameter is case-sensitive and must match the entry name returned by the **List** method exactly.

268.6 Code Example

```
<%
Dim zip
Set zip = Server.CreateObject("G3ZIP")
If zip.Open("/assets/pack.zip") Then
    ' Extract only the documentation
    If zip.ExtractFile("docs/readme.pdf", "/www/help") Then
        Response.Write "Documentation extracted."
    End If
    zip.Close
End If
Set zip = Nothing
%>
```

269 GetInfo Method

269.1 Overview

Retrieves detailed technical metadata for a specific file within the archive in the G3Pix AxonASP G3ZIP library.

269.2 Syntax

```
Set info = zip.GetInfo(fileName)
```

269.3 Parameters and Arguments

- **fileName** (String, Required): The name of the file inside the ZIP archive.

269.4 Return Values

Returns a **Scripting.Dictionary** object containing the following keys: - **Name** (String): The name of the file. - **Size** (Integer): The uncompressed size in bytes. - **PackedSize** (Integer): The compressed size in bytes. - **Modified** (String): The last modification timestamp in RFC3339 format. - **IsDir** (Boolean): Indicates if the entry is a directory.

Returns **Empty** if the file is not found or the object is not in Read mode.

269.5 Remarks

- The object must be in **Read** mode.
- The `fileName` matching is case-insensitive.

269.6 Code Example

```
<%
Dim zip, info
Set zip = Server.CreateObject("G3ZIP")
If zip.Open("/temp/data.zip") Then
    Set info = zip.GetInfo("report.csv")
```

```
    If Not IsEmpty(info) Then
        Response.Write "File: " & info("Name") & "<br>"
        Response.Write "Uncompressed Size: " & info("Size") & " bytes"
    End If
    zip.Close
End If
Set zip = Nothing
%>
```

270 List Method

270.1 Overview

Returns a collection of all file and directory names present in the active archive in the G3Pix AxonASP G3ZIP library.

270.2 Syntax

```
fileNamesArray = zip.List()
```

270.3 Return Values

Returns a **VBArray** (Variant) containing the names of all entries in the archive. If the archive is empty or the object is not in Read mode, it returns an empty array.

270.4 Remarks

- The object must be in **Read** mode.
- Directory entries are typically returned with a trailing slash (e.g., “images/”).

270.5 Code Example

```
<%
Dim zip, files, i
Set zip = Server.CreateObject("G3ZIP")
If zip.Open("/uploads/data.zip") Then
    files = zip.List()
    For i = 0 To UBound(files)
        Response.Write "Entry: " & files(i) & "<br>"
    Next
    zip.Close
End If
Set zip = Nothing
%>
```

271 Open Method

271.1 Overview

Opens an existing ZIP archive file for reading and inspection in the G3Pix AxonASP G3ZIP library.

271.2 Syntax

```
success = zip.Open(archivePath)
```

271.3 Parameters and Arguments

- **archivePath** (String, Required): The path to the ZIP file on the server.

271.4 Return Values

Returns a **Boolean** indicating whether the archive was successfully opened.

271.5 Remarks

- Calling **Open** will close any archive currently managed by the object.
- Once opened, the object enters **Read** mode, enabling methods like **List**, **GetInfo**, and **ExtractAll**.

271.6 Code Example

```
<%
Dim zip
Set zip = Server.CreateObject("G3ZIP")
If zip.Open("/data/source.zip") Then
    Response.Write "Archive has " & zip.Count & " files."
    zip.Close
End If
Set zip = Nothing
%>
```

272 Count Property

272.1 Overview

Returns the total number of files and directories currently contained in the active ZIP archive in the G3Pix AxonASP G3ZIP library.

272.2 Syntax

```
itemCount = zip.Count
```

272.3 Return Values

Returns an **Integer** representing the entry count. If no archive is opened in Read mode, it returns 0.

272.4 Remarks

- This property is only populated when the object is in **Read** mode (initialized via the **Open** method).
- It is read-only.

272.5 Code Example

```
<%
Dim zip
Set zip = Server.CreateObject("G3ZIP")
If zip.Open("/assets/data.zip") Then
    Response.Write "Archive entries: " & zip.Count
    zip.Close
End If
Set zip = Nothing
%>
```

273 Mode Property

273.1 Overview

Indicates the current operating mode of the G3Pix AxonASP G3ZIP library instance.

273.2 Syntax

```
currentMode = zip.Mode
```

273.3 Return Values

Returns a **String**: - **“r”**: The object is in **Read** mode (opened an existing archive). - **“w”**: The object is in **Write** mode (created a new archive). - **““** (Empty String): The object is currently closed.

273.4 Remarks

- This property is read-only.
- The mode is automatically set by the **Open** and **Create** methods.

273.5 Code Example

```
<%
Dim zip
Set zip = Server.CreateObject("G3ZIP")
zip.Create "/temp/test.zip"
If zip.Mode = "w" Then
```

```
    Response.Write "Ready to add files."
End If
Set zip = Nothing
%>
```

274 Path Property

274.1 Overview

Returns the absolute physical path of the archive file currently managed by the G3Pix AxonASP G3ZIP library.

274.2 Syntax

```
fullPath = zip.Path
```

274.3 Return Values

Returns a **String** containing the full path to the ZIP file. Returns an empty string if no archive is active.

274.4 Remarks

- This property is read-only.
- It reflects the resolved path after `Server.MapPath` or relative resolution has been applied by the engine.

274.5 Code Example

```
<%
Dim zip
Set zip = Server.CreateObject("G3ZIP")
zip.Open "data.zip"
Response.Write "Resolved path: " & zip.Path
zip.Close
Set zip = Nothing
%>
```

275 Use G3ZLIB in G3Pix AxonASP

275.1 Overview

G3ZLIB is a compression library that allows you to compress and decompress data using the ZLIB format.

275.2 Syntax

```
Dim obj
Set obj = Server.CreateObject("G3ZLIB")
```

275.3 Parameters and Arguments

- ProgID (String, Required): Use `G3ZLIB` to instantiate this object.
- Member access (Optional): Use the methods and properties provided by the library.

275.4 Return Values

Returns a native object handle for the G3ZLIB library.

275.5 Remarks

- Member names are case-insensitive.
- Runtime validation is enforced by object dispatch logic.
- See the methods and properties pages for member-level coverage.

275.6 Code Example

```
<%
Dim obj
Set obj = Server.CreateObject("G3ZLIB")
Response.Write TypeName(obj)
Set obj = Nothing
%>
```

276 G3ZLIB Methods

276.1 Overview

This page summarizes every method exposed by `G3ZLIB` in G3Pix AxonASP.

276.2 Methods

Method	Returns	Description
`Compress(input [, level])`	Array or Empty	Compresses text or byte-array input and returns compressed bytes as an array; returns Empty when input is missing or compression fails.
`Decompress(input)`	Array or Empty	Decompresses byte-array input and returns decompressed bytes as an array; returns Empty on missing input or decompression failure.

Method	Returns	Description
`DecompressText(input)`	String	Decompresses input and returns UTF-8 text; returns empty string when input is missing or conversion fails.
`CompressMany(items [, level])`	Array	Compresses each item from input array and returns an array of compressed byte arrays; returns empty array when input is missing or any item compression fails.
`DecompressMany(items)`	Array	Decompresses each item from input array and returns an array of decompressed byte arrays.
`CompressFile(sourcePath, destPath [, level])`	Boolean	Compresses one file to destination path. Returns `True` on success; otherwise `False`.
`DecompressFile(sourcePath, destPath)`	Boolean	Decompresses one file to destination path. Returns `True` on success; otherwise `False`.
`Clear()`	Boolean	Clears last error state and returns `True`.

276.3 Remarks

- Instantiate the library with `Server.CreateObject("G3ZLIB")`.
- Method names are case-insensitive.
- Use `LastError` for failure diagnostics.

277 G3ZLIB Properties

277.1 Overview

This page lists the properties exposed by `G3ZLIB`.

277.2 Properties

Property	Access	Type	Description

Property	Access	Type	Description
`LastError`	Read-only	String	Latest error text recorded by compression/decompression operations.

277.3 Remarks

- Instantiate the library with `Server.CreateObject("G3ZLIB")`.

278 Clear G3ZLIB State

278.1 Overview

Clears the current operation context, resetting the internal state and errors of the G3ZLIB object.

278.2 Syntax

```
Call obj.Clear()
```

278.3 Parameters and Arguments

This method takes no arguments.

278.4 Return Values

This method does not return a value.

278.5 Remarks

- Method names are case-insensitive.
- Use this method to reset the `LastError` property and internal buffers before initiating a new independent operation on the same object.

278.6 Code Example

```
<%
Option Explicit
Dim obj
Set obj = Server.CreateObject("G3ZLIB")

Call obj.Clear()

Set obj = Nothing
%>
```

279 Compress Data Using G3ZLIB

279.1 Overview

Compresses text or binary data into a ZLIB compressed format.

279.2 Syntax

```
Dim compressedData
compressedData = obj.Compress(input, level)
```

279.3 Parameters and Arguments

- input (Variant, Required): The string or byte array to compress.
- level (Integer, Optional): The compression level from 1 (fastest) to 9 (best compression). If omitted, it uses the default compression level.

279.4 Return Values

Returns a byte array (Variant array of bytes) containing the compressed data. If the compression fails, it returns Empty and updates the LastError property.

279.5 Remarks

- Method names are case-insensitive.
- Always check if the returned value is Empty to verify if the operation succeeded.

279.6 Code Example

```
<%
Option Explicit
Dim obj, compressedData
Set obj = Server.CreateObject("G3ZLIB")
compressedData = obj.Compress("This is a sample text to compress.", 9)

If IsEmpty(compressedData) Then
    Response.Write "Compression failed: " & obj.LastError
Else
    Response.Write "Compression succeeded."
End If

Set obj = Nothing
%>
```

280 Compress a File Using G3ZLIB

280.1 Overview

Compresses a single source file into an output file.

280.2 Syntax

```
Dim success
success = obj.CompressFile(inputPath, outputPath, level)
```

280.3 Parameters and Arguments

- inputPath (String, Required): The full path to the source file to compress.
- outputPath (String, Required): The destination path for the compressed file.
- level (Integer, Optional): The compression level from 1 (fastest) to 9 (best compression).

280.4 Return Values

Returns a Boolean value. Returns True if the file was compressed and saved successfully; otherwise, returns False and updates the LastError property.

280.5 Remarks

- Method names are case-insensitive.
- The web server must have read permissions for the input file and write permissions for the output path.

280.6 Code Example

```
<%
Option Explicit
Dim obj, success
Set obj = Server.CreateObject("G3ZLIB")

success = obj.CompressFile(Server.MapPath("document.txt"), Server.MapPath
("document.zlib"), 9)

If success Then
    Response.Write "File compressed successfully."
Else
    Response.Write "Error: " & obj.LastError
End If

Set obj = Nothing
%>
```

281 Compress Multiple Files Using G3ZLIB

281.1 Overview

Compresses an array or list of source files into a single output bundle file.

281.2 Syntax

```
Dim success
success = obj.CompressMany(sourcePaths, outputPath, level)
```

281.3 Parameters and Arguments

- sourcePaths (Variant, Required): An array of strings representing the file paths to compress.
- outputPath (String, Required): The destination path for the compressed bundle file.
- level (Integer, Optional): The compression level from 1 (fastest) to 9 (best compression). Defaults to the standard level if omitted.

281.4 Return Values

Returns a Boolean value. Returns True if the files were compressed and saved successfully; otherwise, returns False and updates the LastError property.

281.5 Remarks

- Method names are case-insensitive.
- Ensure that the web server has the appropriate read permissions for the source paths and write permissions for the output directory.

281.6 Code Example

```
<%
Option Explicit
Dim obj, success, files(1)
files(0) = Server.MapPath("file1.txt")
files(1) = Server.MapPath("file2.txt")

Set obj = Server.CreateObject("G3ZLIB")
success = obj.CompressMany(files, Server.MapPath("archive.zlib"), 9)

If success Then
    Response.Write "Files compressed successfully."
Else
    Response.Write "Error: " & obj.LastError
End If

Set obj = Nothing
%>
```

282 Decompress Data Using G3ZLIB

282.1 Overview

Decompresses a ZLIB-compressed byte array back to its original binary representation.

282.2 Syntax

```
Dim originalData
originalData = obj.Decompress(input)
```

282.3 Parameters and Arguments

- input (Variant, Required): The compressed byte array to decompress.

282.4 Return Values

Returns a byte array (Variant array of bytes) containing the decompressed data. If the decompression fails, it returns Empty and updates the LastError property.

282.5 Remarks

- Method names are case-insensitive.
- Use this method when you expect binary data output. For string output, use DecompressText instead.

282.6 Code Example

```
<%
Option Explicit
Dim obj, compressedData, originalData
Set obj = Server.CreateObject("G3ZLIB")

' Assume compressedData is a valid byte array obtained from Compress()
' originalData = obj.Decompress(compressedData)

Set obj = Nothing
%>
```

283 Decompress a File Using G3ZLIB

283.1 Overview

Decompresses a single compressed file back into its original content.

283.2 Syntax

```
Dim success
success = obj.DecompressFile(inputPath, outputPath)
```

283.3 Parameters and Arguments

- inputPath (String, Required): The full path to the compressed source file.
- outputPath (String, Required): The destination path for the decompressed file.

283.4 Return Values

Returns a Boolean value. Returns True if the file was decompressed and saved successfully; otherwise, returns False and updates the LastError property.

283.5 Remarks

- Method names are case-insensitive.
- Ensure the destination path allows write operations.

283.6 Code Example

```
<%
Option Explicit
Dim obj, success
Set obj = Server.CreateObject("G3ZLIB")

success = obj.DecompressFile(Server.MapPath("document.zlib"), Server.MapP
ath("extracted_document.txt"))

If success Then
    Response.Write "File decompressed successfully."
Else
    Response.Write "Error: " & obj.LastError
End If

Set obj = Nothing
%>
```

284 Decompress Multiple Files Using G3ZLIB

284.1 Overview

Decompresses a bundled ZLIB archive back into multiple files in a specified destination directory.

284.2 Syntax

```
Dim success
success = obj.DecompressMany(inputPath, outputFolder)
```

284.3 Parameters and Arguments

- inputPath (String, Required): The path to the compressed bundle file.

- outputFolder (String, Required): The directory where the decompressed files will be saved.

284.4 Return Values

Returns a Boolean value. Returns True if all files were decompressed and saved successfully; otherwise, returns False and updates the LastError property.

284.5 Remarks

- Method names are case-insensitive.
- Ensure that the output folder exists and that the web server has write permissions to it.

284.6 Code Example

```
<%
Option Explicit
Dim obj, success
Set obj = Server.CreateObject("G3ZLIB")

success = obj.DecompressMany(Server.MapPath("archive.zlib"), Server.MapPa
th("output_folder\"))

If success Then
    Response.Write "Archive decompressed successfully."
Else
    Response.Write "Error: " & obj.LastError
End If

Set obj = Nothing
%>
```

285 Decompress Text Using G3ZLIB

285.1 Overview

Decompresses a ZLIB-compressed byte array back to its original text representation.

285.2 Syntax

```
Dim textData
textData = obj.DecompressText(input)
```

285.3 Parameters and Arguments

- input (Variant, Required): The compressed byte array to decompress.

285.4 Return Values

Returns a String containing the decompressed text. If the decompression fails, it returns an empty string and updates the LastError property.

285.5 Remarks

- Method names are case-insensitive.
- This method assumes the original data was a string and automatically handles the text conversion.

285.6 Code Example

```
<%
Option Explicit
Dim obj, textData, compressedData
Set obj = Server.CreateObject("G3ZLIB")

compressedData = obj.Compress("Sample text")
textData = obj.DecompressText(compressedData)

Response.Write textData

Set obj = Nothing
%>
```

286 Get the LastError Property

286.1 Overview

The LastError property is exposed by the G3ZLIB library object and returns the last error message recorded during compression or decompression operations.

286.2 Syntax

```
Dim errorMessage
errorMessage = obj.LastError
```

286.3 Parameters and Arguments

- Getter: Does not take arguments.
- Setter: This property is read-only.

286.4 Return Values

Returns a String containing the description of the last error that occurred. If no error occurred, it returns an empty string.

286.5 Remarks

- Property names are case-insensitive.
- This property is read-only and rejects assignments.

286.6 Code Example

```
<%
Option Explicit
Dim obj, errorMessage
Set obj = Server.CreateObject("G3ZLIB")

' Attempt to decompress invalid data
Call obj.DecompressText("invalid data")

errorMessage = obj.LastError
Response.Write "Last Error: " & errorMessage

Set obj = Nothing
%>
```

287 Use G3TAR in AxonASP

287.1 Overview

G3TAR provides robust TAR archive creation and extraction capabilities. It allows packaging files and folders into an uncompressed tape archive format or extracting existing archives to a local directory.

287.2 Syntax

```
Dim obj
Set obj = Server.CreateObject("G3TAR")
```

287.3 Parameters and Arguments

- ProgID (String, Required): Must initialize with “G3TAR”.

287.4 Return Values

Returns a native object handle for the G3Pix AxonASP G3TAR engine.

287.5 Remarks

- G3TAR works primarily with the uncompressed standard TAR format. For compressed wrappers, combine with G3ZLIB or G3ZSTD operations if needed.
- Always check the returned boolean values from methods and read the LastError property if an operation fails.
- Do not forget to invoke Close after finishing operations to release file handles.

287.6 Code Example

```
<%
Option Explicit
Dim obj
Set obj = Server.CreateObject("G3TAR")
If obj.Create("C:\temp\backup.tar") Then
    obj.AddFile "C:\temp\data.txt", "data.txt"
    obj.Close()
End If
Set obj = Nothing
%>
```

288 G3TAR Methods

288.1 Overview

This page summarizes every method exposed by `G3TAR` in G3Pix AxonASP.

288.2 Methods

Method	Returns	Description
`Create(path)`	Boolean	Creates a TAR archive in write mode. Returns `True` on success; otherwise `False`.
`Open(path)`	Boolean	Opens a TAR archive in read mode and indexes entries. Returns `True` on success; otherwise `False`.
`AddFile(sourcePath [, nameInTar])`	Boolean	Adds one file to archive in write mode. Returns `True` on success; otherwise `False`.
`AddFolder(sourcePath [, nameInTar])`	Boolean	Adds one folder recursively to archive in write mode. Returns `True` on success; otherwise `False`.
`AddFiles(items [, prefix])`	Boolean	Adds multiple items from array/dictionary input to archive in write mode. Returns `True` on success; otherwise `False`.

Method	Returns	Description
`AddText(nameInTar, text)`	Boolean	Adds text content as one TAR entry in write mode. Returns `True` on success; otherwise `False`.
`List()`	Array	Returns an array of TAR entry names.
`ExtractAll(destinationPath)`	Boolean	Extracts all TAR entries in read mode. Returns `True` on success; otherwise `False`.
`ExtractFile(nameInTar, destinationPath)`	Boolean	Extracts one TAR entry in read mode. Returns `True` on success; otherwise `False`.
`GetInfo(nameInTar)`	Scripting.Dictionary or Empty	Returns metadata dictionary for one entry, or Empty when argument is missing or entry is not found.
`Close()`	Boolean	Closes TAR resources and returns `True`.

288.3 Remarks

- Instantiate the library with `Server.CreateObject("G3TAR")`.
- Method names are case-insensitive.
- Use `LastError` for failure diagnostics.

289 G3TAR Properties

289.1 Overview

This page lists the properties exposed by `G3TAR`.

289.2 Properties

Property	Access	Type	Description
`Path`	Read-only	String	Active TAR archive path.
`Mode`	Read-only	String	Current mode (`r` for read or `w` for write).

Property	Access	Type	Description
`Count`	Read-only	Integer	Number of indexed entries in the current TAR context.
`LastError`	Read-only	String	Latest TAR operation error text.

289.3 Remarks

- Instantiate the library with `Server.CreateObject("G3TAR")`.

290 AddFile Method

290.1 Overview

Adds a single local file to the active G3TAR archive.

290.2 Syntax

```
Dim success
success = obj.AddFile(filePath, archiveName)
```

290.3 Parameters and Arguments

- filePath (String, Required): Absolute or relative server-side path pointing to the original file to read.
- archiveName (String, Optional): Target filename or internal directory structure under which the file resides within the TAR archive. If omitted, the base filename is used.

290.4 Return Values

Returns a `Boolean`. Computes to True if the file bytes were flushed to the archive, otherwise False.

290.5 Remarks

- Requires a previous call to the Create method.
- Useful for granular file consolidation across disparate source directories.

290.6 Code Example

```
<%
Option Explicit
Dim obj, success
Set obj = Server.CreateObject("G3TAR")
If obj.Create("C:\temp\output.tar") Then
```

```
    success = obj.AddFile("C:\logs\system.log", "logs/system.log")
    If success Then
        Response.Write "Item appended properly."
    End If
    obj.Close()
End If
Set obj = Nothing
%>
```

291 AddFiles Method

291.1 Overview

Appends a list of specific files to the current G3TAR archive.

291.2 Syntax

```
Dim success
success = obj.AddFiles(sourcePaths, prefix)
```

291.3 Parameters and Arguments

- sourcePaths (Variant, Required): An iterable, such as an Array or a Scripting.Dictionary, containing the paths to append.
- prefix (String, Optional): A common folder path to prepend to the archived names for grouping.

291.4 Return Values

Returns a `Boolean` indicating if all files in the provided structure were successfully appended. True on total success.

291.5 Remarks

- Must be preceded by a successful call to the Create method.
- Useful when selectively archiving files without needing to move them into a consolidated directory first on the server disk.

291.6 Code Example

```
<%
Option Explicit
Dim obj, success, paths(1)
paths(0) = "C:\temp\summary.txt"
paths(1) = "C:\temp\metrics.csv"

Set obj = Server.CreateObject("G3TAR")
If obj.Create("C:\temp\combined.tar") Then
    success = obj.AddFiles(paths, "dataset")
    If success Then
```

```
        Response.Write "Items added!"
    End If
    obj.Close()
End If
Set obj = Nothing
%>
```

292 AddFolder Method

292.1 Overview

Adds a local folder hierarchy to the newly created G3TAR archive.

292.2 Syntax

```
Dim success
success = obj.AddFolder(folderPath, archiveRoot)
```

292.3 Parameters and Arguments

- folderPath (String, Required): The absolute or relative local path to the directory on the server disk.
- archiveRoot (String, Optional): An alternative folder name under which these files will reside inside the TAR archive.

292.4 Return Values

Returns a `Boolean` representing the operation outcome. Returns True if the folder contents were appended successfully, otherwise False.

292.5 Remarks

- Must be called after initializing an archive using the Create method.
- Calling this method automatically cascades into all subdirectories.

292.6 Code Example

```
<%
Option Explicit
Dim obj, success
Set obj = Server.CreateObject("G3TAR")
If obj.Create("C:\temp\backup.tar") Then
    success = obj.AddFolder("C:\temp\reports", "reports_backup")
    If success Then
        Response.Write "Folder added!"
    End If
    obj.Close()
End If
Set obj = Nothing
%>
```

293 AddText Method

293.1 Overview

Adds string content as an inline file into the active G3TAR archive.

293.2 Syntax

```
Dim success
success = obj.AddText(archiveName, text)
```

293.3 Parameters and Arguments

- archiveName (String, Required): Target relative filename where the text will be stored within the TAR.
- text (String, Required): The string to embed as the file body.

293.4 Return Values

Returns a `Boolean` which resolves to True if the text length was fully preserved and saved correctly.

293.5 Remarks

- Requires a current active archive opened via the Create method.
- Allows immediate injection of dynamically generated metadata, manifests, or JSON reports without dumping to the disk first.

293.6 Code Example

```
<%
Option Explicit
Dim obj, success, body
body = "This is a dynamically created file."

Set obj = Server.CreateObject("G3TAR")
If obj.Create("C:\temp\report.tar") Then
    success = obj.AddText("readme.txt", body)
    If success Then
        Response.Write "Text generated straight into archive."
    End If
    obj.Close()
End If
Set obj = Nothing
%>
```

294 Close Method

294.1 Overview

Concludes active session streams alongside removing pending blockades on referenced TAR storage items safely.

294.2 Syntax

```
Dim success
success = obj.Close()
```

294.3 Parameters and Arguments

- Close invocation relies entirely on internal tracking; no parameters required.

294.4 Return Values

Signals operational completion reliably transmitting a `Boolean` equating True indicating termination of locks handled nicely.

294.5 Remarks

- Necessary logic assuring background OS file permissions get properly disposed.
- Abandoned locks could halt consecutive actions targeting similar volumes producing unexpected errors.

294.6 Code Example

```
<%
Option Explicit
Dim obj
Set obj = Server.CreateObject("G3TAR")
If obj.Open("C:\temp\test.tar") Then
    ' File lock assumed...
    obj.Close()
End If
Set obj = Nothing
%>
```

295 Create Method

295.1 Overview

Generates the base file pointer tracking an entirely novel TAR archive mapped structurally onto the server system.

295.2 Syntax

```
Dim success
success = obj.Create(archivePath)
```

295.3 Parameters and Arguments

- archivePath (String, Required): Designated absolute path modeling the storage placement expected upon sequence closure.

295.4 Return Values

Submits a `Boolean` representation validating disk write accessibility and the file sequence lock instantiation correctly. Returns True on success, otherwise it fails yielding False.

295.5 Remarks

- Will aggressively truncate or replace similar file identities encountered explicitly named within the given string array coordinates.
- Expects subsequent interactions populating context data leveraging add components beforehand routing back to Close.

295.6 Code Example

```
<%
Option Explicit
Dim obj, success
Set obj = Server.CreateObject("G3TAR")
success = obj.Create("C:\temp\new_output.tar")
If success Then
    Response.Write "Initiation created."
    obj.Close()
End If
Set obj = Nothing
%>
```

296 ExtractAll Method

296.1 Overview

Reconstructs the complete TAR folder layout natively across a defined destination volume.

296.2 Syntax

```
Dim success
success = obj.ExtractAll(outputFolder)
```

296.3 Parameters and Arguments

- outputFolder (String, Required): Absolute server directory targeting where the complete payload extraction sequence lands.

296.4 Return Values

Returns a `Boolean` equating True if the overall sequence finishes processing without causing integrity blockades or path exceptions.

296.5 Remarks

- Needs a verified session constructed previously targeting the appropriate file handle via Open.
- Automatically handles file access creation permissions when reproducing paths originally saved.

296.6 Code Example

```
<%
Option Explicit
Dim obj, success
Set obj = Server.CreateObject("G3TAR")
If obj.Open("C:\temp\data.tar") Then
    success = obj.ExtractAll("C:\unpacked_volumes")
    If success Then
        Response.Write "Sequence finished."
    End If
    obj.Close()
End If
Set obj = Nothing
%>
```

297 ExtractFile Method

297.1 Overview

Decompresses and pushes one designated file from the connected TAR compilation directly onto the host storage medium.

297.2 Syntax

```
Dim success
success = obj.ExtractFile(archiveName, outputPath)
```

297.3 Parameters and Arguments

- archiveName (String, Required): Target path naming referring directly to the stored inner artifact.

- outputPath (String, Required): Exact endpoint pointer declaring where the output payload becomes solidified onto disk storage.

297.4 Return Values

Dispatches a Boolean expression rendering True for success and False upon error constraints.

297.5 Remarks

- This approach assumes a correct reference initialization on Open usage.
- If folders matching the destination parameters cannot be identified, this sequence will automatically scaffold paths as configured internally.

297.6 Code Example

```
<%
Option Explicit
Dim obj, success
Set obj = Server.CreateObject("G3TAR")
If obj.Open("C:\data\packages\files.tar") Then
    success = obj.ExtractFile("source/test.jpg", "C:\data\extracted\test.
jpg")
    If success Then
        Response.Write "Extract sequence finalized smoothly."
    End If
    obj.Close()
End If
Set obj = Nothing
%>
```

298 GetInfo Method

298.1 Overview

Compiles and maps extensive metadata fields referring to an individual file path stored within the presently loaded TAR structure.

298.2 Syntax

```
Dim dict
Set dict = obj.GetInfo(archiveName)
```

298.3 Parameters and Arguments

- archiveName (String, Required): Target relative naming referring to the file entry inside the TAR file.

298.4 Return Values

Yields a `Scripting.Dictionary` native interface holding extracted fields. Should an anomaly manifest or if no such element occurs, an instantiated blank model will emerge.

298.5 Remarks

- Necessitates pre-opening via the Open invocation.
- Valuable to verify sizing specifications alongside creation timing metadata properties inside the internal container sequence before actual unpacking logic begins.

298.6 Code Example

```
<%
Option Explicit
Dim obj, fileStats
Set obj = Server.CreateObject("G3TAR")
If obj.Open("C:\data\packages\archive.tar") Then
    Set fileStats = obj.GetInfo("docs/readme.txt")
    If fileStats.Count > 0 Then
        Response.Write "Identified item data bounds correctly."
    End If
    Set fileStats = Nothing
    obj.Close()
End If
Set obj = Nothing
%>
```

299 List Method

299.1 Overview

Generates an enumeration of paths that correspond exactly to the current contents found inside the loaded TAR archive.

299.2 Syntax

```
Dim stringArray
stringArray = obj.List()
```

299.3 Parameters and Arguments

- No parameters. Read access must first be set up.

299.4 Return Values

Returns a one-dimensional native array storing String definitions, mapped natively within AxonASP.

299.5 Remarks

- Needs a preceding call referencing an archive via Open.
- Ideal when validating content prior to executing intensive extractions or when generating diagnostic listings.

299.6 Code Example

```
<%
Option Explicit
Dim obj, files, i
Set obj = Server.CreateObject("G3TAR")
If obj.Open("C:\data\packages\archive.tar") Then
    files = obj.List()
    For i = LBound(files) To UBound(files)
        Response.Write files(i) & "<br>"
    Next
    obj.Close()
End If
Set obj = Nothing
%>
```

300 Open Method

300.1 Overview

Accesses an established TAR archive on the server to read or extract its contents.

300.2 Syntax

```
Dim success
success = obj.Open(archivePath)
```

300.3 Parameters and Arguments

- archivePath (String, Required): Exact system location indicating the existing TAR file on the filesystem.

300.4 Return Values

Returns a `Boolean` representing whether the operation locked the file properly. Returns True if successfully mounted.

300.5 Remarks

- Must be combined with a paired extraction routine or metadata gathering phase followed immediately by a manual invocation of Close.
- Do not use this method on archives concurrently accessed through other means on the local disk.

300.6 Code Example

```
<%
Option Explicit
Dim obj, success
Set obj = Server.CreateObject("G3TAR")
success = obj.Open("C:\data\packages\archive.tar")
If success Then
    Response.Write "Successfully connected."
    obj.Close()
End If
Set obj = Nothing
%>
```

301 Count Property

301.1 Overview

Gets the total number of entries within the active G3TAR archive.

301.2 Syntax

```
Dim entryCount
entryCount = obj.Count
```

301.3 Parameters and Arguments

- Getter: None.

301.4 Return Values

Returns an `Integer` representing the total files and folders tracked.

301.5 Remarks

- This property is read-only.
- Returns zero if no archive is open.

301.6 Code Example

```
<%
Option Explicit
Dim obj, entryCount
Set obj = Server.CreateObject("G3TAR")
If obj.Open("C:\temp\archive.tar") Then
    entryCount = obj.Count
    Response.Write entryCount
End If
obj.Close()
Set obj = Nothing
%>
```

302 LastError Property

302.1 Overview

Gets the description of the last error encountered during a G3TAR operation.

302.2 Syntax

```
Dim errorMsg
errorMsg = obj.LastError
```

302.3 Parameters and Arguments

- Getter: None.

302.4 Return Values

Returns a `String` detailing the most recent internal operation error.

302.5 Remarks

- This property is read-only.
- Use this to inspect failures instead of relying solely on Boolean returns.

302.6 Code Example

```
<%
Option Explicit
Dim obj, errorMsg
Set obj = Server.CreateObject("G3TAR")
If Not obj.Open("C:\temp\missing.tar") Then
    errorMsg = obj.LastError
    Response.Write errorMsg
End If
Set obj = Nothing
%>
```

303 Mode Property

303.1 Overview

Gets the current operation mode of the G3TAR archive.

303.2 Syntax

```
Dim currentMode
currentMode = obj.Mode
```

303.3 Parameters and Arguments

- Getter: None.

303.4 Return Values

Returns a `String` representing the archive mode.

303.5 Remarks

- This property is read-only.
- Usually reflects read or write modes depending on initialization.

303.6 Code Example

```
<%
Option Explicit
Dim obj, currentMode
Set obj = Server.CreateObject("G3TAR")
If obj.Open("C:\temp\archive.tar") Then
    currentMode = obj.Mode
    Response.Write currentMode
End If
obj.Close()
Set obj = Nothing
%>
```

304 Path Property

304.1 Overview

Gets the active file path associated with the G3TAR archive.

304.2 Syntax

```
Dim filePath
filePath = obj.Path
```

304.3 Parameters and Arguments

- Getter: None.

304.4 Return Values

Returns a `String` containing the absolute path of the current archive.

304.5 Remarks

- This property is read-only.
- Available only after opening or creating an archive.

304.6 Code Example

```
<%
Option Explicit
Dim obj, filePath
```

```
Set obj = Server.CreateObject("G3TAR")
If obj.Open("C:\temp\archive.tar") Then
    filePath = obj.Path
    Response.Write filePath
End If
obj.Close()
Set obj = Nothing
%>
```

305 Use the G3ZSTD Library

305.1 Overview

The **G3ZSTD** library provides high-performance data compression and decompression services for G3Pix AxonASP applications using the Zstandard (Zstd) algorithm. It is optimized for both speed and compression ratio, offering a wide range of compression levels. The library supports in-memory processing of strings and byte arrays, batch processing of arrays, and streaming file compression to minimize memory footprint.

305.2 Syntax

To instantiate the library, use the following syntax:

```
Set zstd = Server.CreateObject("G3ZSTD")
```

305.3 Prerequisites

No external dependencies are required. The G3ZSTD library is a built-in native component of the G3Pix AxonASP environment.

305.4 How it Works

The G3ZSTD object provides a stateless interface for most compression operations but maintains internal state for the **Level** configuration and the **LastError** property. - **Levels**: Compression levels range from -5 (fastest) to 22 (highest compression). The default level is 3. - **Data Types**: The library seamlessly handles VBScript strings and byte arrays (VBArray of bytes). - **File Streaming**: Methods like **CompressFile** and **DecompressFile** use streaming I/O to process large files without loading the entire content into memory, ensuring system stability. - **Resource Management**: While the object is managed by the AxonASP garbage collector, the **Clear** method can be used to explicitly release internal encoder/decoder buffers.

305.5 API Reference

305.5.1 Methods

- **Clear**: Releases internal buffers and resets the error state.

- **Compress**: Compresses a string or byte array into a Zstd-encoded byte array.
- **CompressFile**: Compresses a file from disk and saves the result to a new file.
- **CompressMany**: Performs batch compression on an array of inputs.
- **Decompress**: Decompresses a Zstd-encoded byte array into its original byte array.
- **DecompressFile**: Decompresses a Zstd file from disk to its original state.
- **DecompressMany**: Performs batch decompression on an array of encoded inputs.
- **DecompressText**: Decompresses a Zstd-encoded byte array directly into a UTF-8 string.
- **SetLevel**: Configures the default compression level for the object instance.

305.5.2 Properties

- **LastError**: Returns the description of the most recent error encountered.
- **Level**: Returns the currently configured default compression level.

305.6 Code Example

The following example demonstrates how to compress a string and luego decompress it back to text.

```
<%
Dim zstd, original, compressed, recovered
Set zstd = Server.CreateObject("G3ZSTD")

' Original data
original = "AxonASP high-performance compression test data."

' Compress data at level 9
compressed = zstd.Compress(original, 9)

' Decompress directly to string
recovered = zstd.DecompressText(compressed)

If recovered = original Then
    Response.Write "Compression and Decompression Successful"
Else
    Response.Write "Operation Failed: " & zstd.LastError
End If

Set zstd = Nothing
%>
```

306 G3ZSTD Methods

306.1 Overview

This page summarizes the methods available in the G3ZSTD library for Zstandard compression and decompression in G3Pix AxonASP.

306.2 Methods Reference

Method	Returns	Description
SetLevel	Boolean	Sets the default compression level for the instance.
Compress	Byte Array or Empty	Compresses a string or byte array payload and returns compressed bytes. Returns Empty when input is invalid or compression fails.
Decompress	Byte Array or Empty	Decompresses a compressed payload and returns original bytes. Returns Empty when payload is invalid or decompression fails.
DecompressText	String	Decompresses a payload and converts the result to UTF-8 text. Returns an empty string on failure.
CompressMany	Array of Byte Arrays	Compresses each item from an input array. Returns an empty array when one of the items is invalid.
DecompressMany	Array of Byte Arrays	Decompresses each item from an input array. Returns an empty array when one of the items is invalid.
CompressFile	Boolean	Compresses a source file to a target file path.

Method	Returns	Description
DecompressFile	Boolean	Decompresses a source file to a target file path.
Clear	Boolean	Releases internal encoder and decoder state and clears the last error.

306.3 Remarks

- Method names are case-insensitive.
- Valid compression levels are from -5 to 22.
- After operational errors, inspect LastError.

307 G3ZSTD Properties

307.1 Overview

This page summarizes the properties available in the G3ZSTD library for runtime inspection.

307.2 Properties Reference

Property	Access	Type	Description
LastError	Read-only	String	Stores the most recent operation error message.
Level	Read-only	Integer	Gets the current default compression level.

307.3 Remarks

- Property names are case-insensitive.
- Use SetLevel to change the compression level.

308 Clear Method

308.1 Overview

Clears the internal compression and decompression context of the G3Pix AxonASP G3ZSTD object. This method closes any active encoder or decoder resources and resets the last error message.

308.2 Syntax

```
result = obj.Clear()
```

308.3 Parameters and Arguments

This method does not accept any parameters.

308.4 Return Values

Returns a Boolean value: - **True**: The internal state was successfully cleared.

308.5 Remarks

- Use this method to free up system resources if the object is being reused for multiple unrelated operations.
- This method is also aliased as `Initialize` and `Dispose`.
- If an error occurs during clearing, a runtime error is raised.

308.6 Code Example

```
<%
Option Explicit
Dim objZstd
Set objZstd = Server.CreateObject("G3ZSTD")

' Perform operations...

' Clear the internal state
If objZstd.Clear() Then
    Response.Write "Context cleared successfully."
End If

Set objZstd = Nothing
%>
```

309 Compress Method

309.1 Overview

Compresses a string or a byte array using the Zstandard (zstd) algorithm. This G3Pix AxonASP method provides high-performance compression with adjustable levels.

309.2 Syntax

```
result = obj.Compress(input [, level])
```

309.3 Parameters and Arguments

- **input**: A String or a VBArray of bytes containing the data to be compressed.
- **level** (Optional): An Integer specifying the compression level. The range is -5 to 22. If omitted, the default level (currently 3) is used.

309.4 Return Values

Returns a VBArray of bytes containing the compressed data. If an error occurs, it returns Empty.

309.5 Remarks

- The compression level can range from -5 (fastest) to 22 (highest compression ratio).
- Higher compression levels consume more memory and CPU.
- Input validation ensures that the input is either a string or a valid array of bytes (0-255).
- If the compression level is invalid, a runtime error is raised.

309.6 Code Example

```
<%
Option Explicit
Dim objZstd, compressedData, sourceString
sourceString = "This is some sample text to compress using G3Pix AxonASP
G3ZSTD."

Set objZstd = Server.CreateObject("G3ZSTD")

' Compress using default level
compressedData = objZstd.Compress(sourceString)

If Not IsEmpty(compressedData) Then
    Response.Write "Compressed data size: " & UBound(compressedData) + 1
& " bytes."
Else
    Response.Write "Compression failed: " & objZstd.LastError
End If

Set objZstd = Nothing
%>
```

310 CompressFile Method

310.1 Overview

Compresses a source file and writes the result to a target file using the Zstandard (zstd) algorithm. This G3Pix AxonASP method uses streaming to keep memory usage low even for large files.

310.2 Syntax

```
result = obj.CompressFile(sourcePath, targetPath [, level])
```

310.3 Parameters and Arguments

- **sourcePath**: A String specifying the path to the source file.
- **targetPath**: A String specifying the path where the compressed file will be created.
- **level** (Optional): An Integer specifying the compression level. The range is -5 to 22. If omitted, the default level is used.

310.4 Return Values

Returns a Boolean value: - **True**: The file was successfully compressed. - **False**: An error occurred during compression (e.g., file not found, permission denied).

310.5 Remarks

- Both source and target paths are resolved relative to the web root and must stay within the sandbox.
- This method automatically creates the target directory if it does not exist.
- Streaming ensures that memory consumption is minimal regardless of the file size.
- Errors are logged to the `LastError` property and raise runtime exceptions.

310.6 Code Example

```
<%
Option Explicit
Dim objZstd
Set objZstd = Server.CreateObject("G3ZSTD")

' Compress a file with level 9
If objZstd.CompressFile("data/log.txt", "data/log.txt.zst", 9) Then
    Response.Write "File compressed successfully."
Else
    Response.Write "Error: " & objZstd.LastError
End If

Set objZstd = Nothing
%>
```

311 CompressMany Method

311.1 Overview

Compresses an array of strings or byte arrays using the Zstandard (zstd) algorithm. This G3Pix AxonASP method allows for batch compression of multiple payloads efficiently.

311.2 Syntax

```
resultArray = obj.CompressMany(inputArray [, level])
```

311.3 Parameters and Arguments

- **inputArray**: A VBArray containing one or more elements. Each element must be a String or a VBArray of bytes.
- **level** (Optional): An Integer specifying the compression level. The range is -5 to 22. If omitted, the default level is used.

311.4 Return Values

Returns a VBArray containing several VBArrays of bytes, where each sub-array corresponds to the compressed data of the input items.

311.5 Remarks

- This method provides efficient batch processing for multiple payloads.
- If any input item fails validation or compression, an empty array or error may be returned depending on the context.
- The compression level remains constant for all items in the batch.
- Errors during batch processing are logged to the `LastError` property.

311.6 Code Example

```
<%
Option Explicit
Dim objZstd, payloads, results, i
payloads = Array("Item 1", "Item 2", "Item 3")

Set objZstd = Server.CreateObject("G3ZSTD")

' Batch compress with default level
results = objZstd.CompressMany(payloads)

For i = 0 To UBound(results)
    Response.Write "Item " & i & " compressed size: " & UBound(results(i))
 + 1 & " bytes.<br>"
Next

Set objZstd = Nothing
%>
```

312 Decompress Method

312.1 Overview

Decompresses a Zstandard (zstd) compressed payload (string or byte array) and returns the original data as a byte array.

312.2 Syntax

```
result = obj.Decompress(input)
```

312.3 Parameters and Arguments

- **input**: A String or a VBArray of bytes containing the Zstandard compressed data.

312.4 Return Values

Returns a VBArray of bytes containing the decompressed data. If an error occurs, it returns Empty.

312.5 Remarks

- The input must be a valid Zstandard frame.
- This G3Pix AxonASP method handles all data types by normalizing them to a byte stream before decompression.
- If decompression fails (e.g., due to corrupt data or invalid format), a runtime error is raised.

312.6 Code Example

```
<%
Option Explicit
Dim objZstd, compressedData, originalData
' Assuming compressedData is obtained from a source or previous operation
Set objZstd = Server.CreateObject("G3ZSTD")

originalData = objZstd.Decompress(compressedData)

If Not IsEmpty(originalData) Then
    Response.Write "Original data size: " & UBound(originalData) + 1 & "
bytes."
Else
    Response.Write "Decompression failed."
End If

Set objZstd = Nothing
%>
```

313 DecompressFile Method

313.1 Overview

Decompresses a Zstandard (zstd) compressed file and writes the result to a target file. This G3Pix AxonASP method uses streaming to keep memory usage low even for large files.

313.2 Syntax

```
result = obj.DecompressFile(sourcePath, targetPath)
```

313.3 Parameters and Arguments

- **sourcePath**: A String specifying the path to the compressed (.zst) file.
- **targetPath**: A String specifying the path where the decompressed file will be created.

313.4 Return Values

Returns a Boolean value: - **True**: The file was successfully decompressed. - **False**: An error occurred during decompression (e.g., file not found, corrupt data).

313.5 Remarks

- Both source and target paths are resolved relative to the web root and must stay within the sandbox.
- This method automatically creates the target directory if it does not exist.
- Streaming ensures that memory consumption is minimal regardless of the file size.
- Errors are logged to the `LastError` property and raise runtime exceptions.

313.6 Code Example

```
<%
Option Explicit
Dim objZstd
Set objZstd = Server.CreateObject("G3ZSTD")

' Decompress a file
If objZstd.DecompressFile("data/log.txt.zst", "data/log_restored.txt") Th
en
    Response.Write "File decompressed successfully."
Else
    Response.Write "Error: " & objZstd.LastError
End If

Set objZstd = Nothing
%>
```

314 DecompressMany Method

314.1 Overview

Decompresses an array of Zstandard (zstd) compressed payloads (strings or byte arrays). This G3Pix AxonASP method allows for efficient batch decompression of multiple datasets.

314.2 Syntax

```
resultArray = obj.DecompressMany(inputArray)
```

314.3 Parameters and Arguments

- **inputArray**: A VBArray containing one or more elements. Each element must be a String or a VBArray of bytes containing a Zstandard compressed frame.

314.4 Return Values

Returns a VBArray containing several VBArrays of bytes, where each sub-array corresponds to the decompressed data of the input items.

314.5 Remarks

- Each item in the input array must be a valid Zstandard frame.
- The method processes items in sequence and returns an array of the same length as the input.
- If any item fails decompression, the overall operation may result in an empty array or raise a runtime error.
- Errors during batch processing are logged to the `LastError` property.

314.6 Code Example

```
<%
Option Explicit
Dim objZstd, compressedItems, decompressedItems, i
' Assuming compressedItems is an array of compressed data

Set objZstd = Server.CreateObject("G3ZSTD")

' Batch decompress
decompressedItems = objZstd.DecompressMany(compressedItems)

For i = 0 To UBound(decompressedItems)
    Response.Write "Item " & i & " decompressed size: " & UBound(decompre
ssedItems(i)) + 1 & " bytes.<br>"
Next
```

```
Set objZstd = Nothing
%>
```

315 DecompressText Method

315.1 Overview

Decompresses a Zstandard (zstd) compressed payload (string or byte array) and returns the original data as a UTF-8 string.

315.2 Syntax

```
resultString = obj.DecompressText(input)
```

315.3 Parameters and Arguments

- **input**: A String or a VBArray of bytes containing the Zstandard compressed data.

315.4 Return Values

Returns a String containing the decompressed UTF-8 text. If an error occurs, it returns an empty string.

315.5 Remarks

- This method is an alias for `DecompressString`.
- It expects the original uncompressed data to be a valid UTF-8 string.
- If decompression fails or the data is not valid UTF-8, it returns an empty string and logs the error to the `LastError` property.

315.6 Code Example

```
<%
Option Explicit
Dim objZstd, compressedData, originalText
' Assuming compressedData is obtained from a source or previous operation
Set objZstd = Server.CreateObject("G3ZSTD")

originalText = objZstd.DecompressText(compressedData)

If originalText <> "" Then
    Response.Write "Restored text: " & originalText
Else
    Response.Write "Decompression failed or returned empty text."
End If

Set objZstd = Nothing
%>
```

316 SetLevel Method

316.1 Overview

Sets the default compression level for the G3Pix AxonASP G3ZSTD object. This level is used for all subsequent compression operations that do not explicitly specify a level.

316.2 Syntax

```
result = obj.SetLevel(level)
```

316.3 Parameters and Arguments

- **level**: An Integer specifying the compression level. The range is -5 (fastest, lower ratio) to 22 (slowest, highest ratio).

316.4 Return Values

Returns a Boolean value: - **True**: The compression level was successfully updated. - **False**: An invalid level was provided (outside the -5 to 22 range).

316.5 Remarks

- This method is an alias for `SetCompressionLevel`.
- Setting a new level will re-initialize the internal encoder resource.
- The default compression level is 3.
- If an invalid level is provided, the current level is maintained and a runtime error is raised.

316.6 Code Example

```
<%
Option Explicit
Dim objZstd
Set objZstd = Server.CreateObject("G3ZSTD")

' Change default compression level to 12
If objZstd.SetLevel(12) Then
    Response.Write "Default compression level set to " & objZstd.Level
End If

' Use the new default level for compression
Dim compressed
compressed = objZstd.Compress("Test Data")

Set objZstd = Nothing
%>
```

317 LastError Property

317.1 Overview

Returns a String describing the last error that occurred during a Zstandard (zstd) operation in the G3Pix AxonASP G3ZSTD object.

317.2 Syntax

```
errorString = obj.LastError
```

317.3 Parameters and Arguments

This property is read-only and does not accept any parameters.

317.4 Return Values

Returns a String containing the error message associated with the last failed operation. If no error has occurred, it returns an empty string.

317.5 Remarks

- This property is useful for diagnostic purposes when a method (like `CompressFile`) returns `False`.
- The error message is reset when `Clear` is called or when a new operation succeeds.
- Runtime exceptions also contain these messages.

317.6 Code Example

```
<%
Option Explicit
Dim objZstd
Set objZstd = Server.CreateObject("G3ZSTD")

' Perform an operation that might fail
If Not objZstd.CompressFile("missing_file.txt", "target.zst") Then
    Response.Write "Operation failed with error: " & objZstd.LastError
End If

Set objZstd = Nothing
%>
```

318 Level Property

318.1 Overview

Gets the current default compression level for the G3Pix AxonASP G3ZSTD object.

318.2 Syntax

```
currentLevel = obj.Level
```

318.3 Parameters and Arguments

This property is read-only. To set the level, use the `SetLevel` method.

318.4 Return Values

Returns an Integer representing the current compression level. The range is -5 to 22.

318.5 Remarks

- This property is an alias for `CompressionLevel`.
- The default compression level is 3.
- This level is used for all compression methods unless a level is explicitly provided as an argument.
- Zstandard levels: -5 (fastest) up to 22 (best compression).

318.6 Code Example

```
<%
Option Explicit
Dim objZstd
Set objZstd = Server.CreateObject("G3ZSTD")

' Read default level
Response.Write "Default level is " & objZstd.Level

' Change level
objZstd.SetLevel(15)
Response.Write "New level is " & objZstd.Level

Set objZstd = Nothing
%>
```

319 Use the G3FC Archive Library in G3Pix AxonASP

319.1 Overview

Use the G3FC library to create, inspect, search, and extract G3FC archives in G3Pix AxonASP. This page provides a concise API summary for object creation, available methods, and property behavior.

319.2 Syntax

```
Set fc = Server.CreateObject("G3FC")
```

319.3 Parameters and Arguments

- ProgID (String, Required): `G3FC`.
- Object variable (Object, Required): Receives the instantiated native G3FC object handle.
- Method arguments (Depends on method): Each method validates argument count and argument types at runtime.

319.4 Return Values

- `Server.CreateObject("G3FC")` returns a native object handle bound to the G3FC implementation.
- If object creation fails, the call returns `Empty` and `Server.GetLastError` contains the failure details.

319.5 Remarks

- **Object creation is case-insensitive for supported ProgIDs.** `G3FC`, `g3fc`, and mixed-case forms resolve to the same native implementation.
- **Unsupported ProgIDs fall through to host-level CreateObject handling.** If the host cannot create the object, AxonASP raises `AxonASP cannot create object`.
- **Compatibility aliases are mapped to native AxonASP implementations where available.** For G3FC specifically, use the primary ProgID `G3FC`.
- **Method names are case-insensitive.**
- G3FC exposes no dedicated properties. Property-style access is routed to method dispatch.

319.6 Code Example

```
<%
Option Explicit

Dim fc
Set fc = Server.CreateObject("G3FC")

If IsObject(fc) Then
    Response.Write "G3FC object created successfully."
Else
    Response.Write "Failed to create G3FC object."
End If

Set fc = Nothing
%>
```

319.7 Methods Reference

Method	Returns	Description
Create	Boolean	Returns `True` when the archive is created successfully; returns `False` when required arguments are missing, path resolution fails, or archive creation fails.
Extract	Boolean	Returns `True` when all archive entries are extracted to the target directory; returns `False` when arguments are invalid or extraction fails.
List	Array of `Scripting.Dictionary` or Empty	Returns an Array where each item is a `Scripting.Dictionary` describing one archive entry (for example, path and size metadata). Returns `Empty` when arguments are invalid or archive reading fails.
Info	Boolean	Returns `True` when metadata export to the output file completes; returns `False` when arguments are invalid or metadata export fails.
Find	Array of `Scripting.Dictionary` or Empty	Returns an Array where each item is a `Scripting.Dictionary` for an entry matched by substring or regular expression. Returns `Empty` when arguments are invalid or search/read fails.
ExtractSingle	Boolean	Returns `True` when the

Method	Returns	Description
		requested single entry is extracted to the target path; returns `False` when arguments are invalid, the entry does not exist, or extraction fails.

319.8 Properties Reference

Property	Access	Type	Description
None	Not applicable	Not applicable	G3FC does not expose dedicated properties. Use method calls for all operations.

319.9 API Reference

- Object creation: `Server.CreateObject("G3FC")`
- Method aliases: `ExtractSingle` also accepts `extract-single` and `extract_single`
- Member resolution: case-insensitive dispatch for methods and property-style calls

320 G3FC Methods

320.1 Overview

This page summarizes every method exposed by G3FC in G3Pix AxonASP.

320.2 Methods Reference

Method	Returns	Description
Create(archivePath, sourcePaths [, password] [, options])	Boolean	Returns True when the archive is created successfully. Returns False on invalid arguments, invalid output path resolution, empty resolved source set, or archive creation failure.
Extract(archivePath, outputFolder [, password])	Boolean	Returns True when all entries are extracted

Method	Returns	Description
		successfully. Returns False on invalid arguments, invalid path resolution, or extraction failure.
List(archivePath [, password] [, unit] [, details])	Array of Scripting.Dictionary or Empty	Returns a zero-based Array of Dictionary entries with Path, Size, FormattedSize, and Type keys. When details=True, each Dictionary also includes Permissions, CreationTime, and Checksum. Returns Empty when arguments are invalid, path resolution fails, or index read fails.
Info(archivePath, outputFilePath [, password])	Boolean	Returns True when archive metadata is exported to the output file. Returns False on invalid arguments, invalid path resolution, or export failure.
Find(archivePath, pattern [, password] [, useRegex])	Array of Scripting.Dictionary or Empty	Returns a zero-based Array of Dictionary entries for matches, each with Path and Size keys. Returns Empty when arguments are invalid, path resolution fails, or index read fails.
ExtractSingle(archivePath, entryPath, outputFilePath [, password])	Boolean	Returns True when one specific entry is extracted successfully. Returns False on invalid arguments, invalid path resolution, missing entry, or extraction failure.

320.3 Remarks

- Method names are case-insensitive.
- ExtractSingle also accepts aliases extract-single and extract_single.
- Array-returning methods return Empty on validation/read failures and do not return partial scalar fallback values.

321 G3FC Properties

321.1 Overview

This page summarizes property behavior for G3FC in G3Pix AxonASP.

321.2 Properties Reference

Property	Access	Type	Description
None	Not applicable	Not applicable	G3FC does not define persistent object properties. Any property-style access is dispatched as a method name lookup. Unknown members return Empty.

321.3 Remarks

- Use the methods listed on the methods page for all operations.

322 Create Method

322.1 Overview

Create a new G3FC archive from one file path or multiple source paths.

322.2 Syntax

```
result = fc.Create(archivePath, sourcePaths [, password] [, options])
```

322.3 Parameters and Arguments

- archivePath (String, Required): Output `.g3fc` file path. The path is resolved through the current ASP host root.
- sourcePaths (String or Array, Required): Source input. Pass one path as String, or pass a VBScript Array of paths.

- password (String, Optional): Read password written into the archive configuration. When non-empty, encryption mode is enabled.
- options (Scripting.Dictionary, Optional): Configuration Dictionary. Supported keys:
    - `CompressionLevel` (Integer): Zstandard compression level. Default is `6`.
    - `GlobalCompression` (Boolean): If `True`, keeps file blocks uncompressed and uses global compression mode.
    - `FECLevel` (Integer): Forward error correction level. Values greater than `0` enable FEC.
    - `SplitSize` (String): Segment size in `MB` or `GB` format (example: `100MB`, `1GB`).

322.4 Return Values

- Returns `True` when archive creation completes successfully.
- Returns `False` when any required argument is missing, output path resolution fails, no valid source files are resolved, or archive creation fails.

322.5 Remarks

- Method names are case-insensitive.
- Invalid source paths are skipped during source resolution.
- If no valid source file remains after resolution, the method returns `False`.
- Runtime failures raise an internal VBScript error and the method still returns `False`.

322.6 Code Example

```
<%
Option Explicit
Dim fc, options, ok
Set fc = Server.CreateObject("G3FC")
Set options = Server.CreateObject("Scripting.Dictionary")

Call options.Add("CompressionLevel", 9)
Call options.Add("GlobalCompression", False)
Call options.Add("FECLevel", 10)
Call options.Add("SplitSize", "100MB")

ok = fc.Create("/sandbox/archive.g3fc", Array("/sandbox/a.txt", "/sandbox
/b.txt"), "AxonPass", options)

If ok Then
    Response.Write "Archive created successfully."
Else
    Response.Write "Archive creation failed."
```

```
End If

Set options = Nothing
Set fc = Nothing
%>
```

323 Extract Method

323.1 Overview

Extract all entries from a G3FC archive into a target directory.

323.2 Syntax

```
result = fc.Extract(archivePath, outputFolder [, password])
```

323.3 Parameters and Arguments

- archivePath (String, Required): Source `.g3fc` file path.
- outputFolder (String, Required): Destination directory path.
- password (String, Optional): Archive password used when the archive is encrypted.

323.4 Return Values

- Returns `True` when extraction completes successfully.
- Returns `False` when arguments are missing, path resolution fails, or extraction fails.

323.5 Remarks

- Method names are case-insensitive.
- Runtime extraction failures raise an internal VBScript error and the method still returns `False`.

323.6 Code Example

```
<%
Option Explicit
Dim fc, ok
Set fc = Server.CreateObject("G3FC")

ok = fc.Extract("/sandbox/archive.g3fc", "/sandbox/extracted", "AxonPass")

If ok Then
    Response.Write "Archive extracted successfully."
Else
    Response.Write "Archive extraction failed."
End If
```

```
Set fc = Nothing
%>
```

324 ExtractSingle Method

324.1 Overview

Extract one specific file entry from a G3FC archive.

324.2 Syntax

```
result = fc.ExtractSingle(archivePath, entryPath, outputFilePath [, passw
ord])
```

324.3 Parameters and Arguments

- archivePath (String, Required): Source `.g3fc` file path.
- entryPath (String, Required): Internal archive entry path to extract.
- outputFilePath (String, Required): Destination file path for the extracted entry.
- password (String, Optional): Archive password for encrypted archives.

324.4 Return Values

- Returns `True` when the target entry is extracted successfully.
- Returns `False` when required arguments are missing, path resolution fails, the entry does not exist, or extraction fails.

324.5 Remarks

- Method names are case-insensitive.
- This method also accepts `extract-single` and `extract_single` aliases.
- Runtime extraction failures raise an internal VBScript error and the method still returns `False`.

324.6 Code Example

```
<%
Option Explicit
Dim fc, ok
Set fc = Server.CreateObject("G3FC")

ok = fc.ExtractSingle("/sandbox/archive.g3fc", "docs/readme.txt", "/sandb
ox/readme.txt", "AxonPass")

If ok Then
    Response.Write "Single entry extracted successfully."
Else
    Response.Write "Single entry extraction failed."
End If
```

```
Set fc = Nothing
%>
```

325 Find Method

325.1 Overview

Search archive entries by substring or regular expression.

325.2 Syntax

```
matches = fc.Find(archivePath, pattern [, password] [, useRegex])
```

325.3 Parameters and Arguments

- archivePath (String, Required): Source `.g3fc` file path.
- pattern (String, Required): Match expression applied to entry paths.
- password (String, Optional): Archive password for encrypted archives.
- useRegex (Boolean, Optional):
  - `False` (default): uses case-insensitive substring matching.
  - `True`: attempts case-insensitive regular expression matching.

325.4 Return Values

- Returns a zero-based Array of `Scripting.Dictionary` objects on success.
- Each Dictionary includes exactly two keys: `Path` (String) and `Size` (Integer).
- Returns an empty Array when no entries match.
- Returns `Empty` when required arguments are missing, path resolution fails, or archive index reading fails.

325.5 Remarks

- Method names are case-insensitive.
- Regex matching is case-insensitive.
- If `useRegex=True` and the expression fails to compile, matching falls back to case-insensitive substring behavior.
- Runtime read failures raise an internal VBScript error and the method returns `Empty`.

325.6 Code Example

```
<%
Option Explicit
Dim fc, matches, i, item
Set fc = Server.CreateObject("G3FC")

matches = fc.Find("/sandbox/archive.g3fc", "\\.txt$", "AxonPass", True)
```

```
If IsArray(matches) Then
    For i = LBound(matches) To UBound(matches)
        Set item = matches(i)
        Response.Write item.Item("Path") & "<br>"
        Set item = Nothing
    Next
Else
    Response.Write "Find failed."
End If

Set fc = Nothing
%>
```

326 Info Method

326.1 Overview

Export archive metadata from a G3FC file into an output file.

326.2 Syntax

```
result = fc.Info(archivePath, outputFilePath [, password])
```

326.3 Parameters and Arguments

- archivePath (String, Required): Source `.g3fc` file path.
- outputFilePath (String, Required): Destination file path that receives the exported metadata.
- password (String, Optional): Archive password for encrypted archives.

326.4 Return Values

- Returns `True` when metadata export succeeds.
- Returns `False` when required arguments are missing, path resolution fails, or export fails.

326.5 Remarks

- Method names are case-insensitive.
- Runtime export failures raise an internal VBScript error and the method still returns `False`.

326.6 Code Example

```
<%
Option Explicit
Dim fc, ok
Set fc = Server.CreateObject("G3FC")
```

```
ok = fc.Info("/sandbox/archive.g3fc", "/sandbox/archive-info.txt", "AxonPass")

If ok Then
    Response.Write "Archive info exported."
Else
    Response.Write "Archive info export failed."
End If

Set fc = Nothing
%>
```

327 List Method

327.1 Overview

List entries stored in a G3FC archive.

327.2 Syntax

```
entries = fc.List(archivePath [, password] [, unit] [, details])
```

327.3 Parameters and Arguments

- archivePath (String, Required): Source `.g3fc` file path.
- password (String, Optional): Archive password for encrypted archives.
- unit (String, Optional): Formatted size unit used by `FormattedSize`. Supported values are `TB`, `GB`, `MB`, and `KB`. Default is `KB`. Any other value returns sizes as `B` text.
- details (Boolean, Optional): When `True`, includes extended metadata fields in each result item. Default is `False`.

327.4 Return Values

- Returns a zero-based Array of `Scripting.Dictionary` objects when listing succeeds.
- Each Dictionary always includes: `Path` (String), `Size` (Integer), `FormattedSize` (String), and `Type` (String).
- When `details=True`, each Dictionary also includes: `Permissions` (String in octal format), `CreationTime` (RFC3339 String), and `Checksum` (uppercase 8-digit hexadecimal String).
- Returns `Empty` when required arguments are missing, path resolution fails, or archive index reading fails.

327.5 Remarks

- Method names are case-insensitive.

- Runtime read failures raise an internal VBScript error and the method returns `Empty`.

327.6 Code Example

```
<%
Option Explicit
Dim fc, entries, i, item
Set fc = Server.CreateObject("G3FC")

entries = fc.List("/sandbox/archive.g3fc", "AxonPass", "MB", True)

If IsArray(entries) Then
    For i = LBound(entries) To UBound(entries)
        Set item = entries(i)
        Response.Write item.Item("Path") & " - " & item.Item("FormattedSi
ze") & "<br>"
        Set item = Nothing
    Next
Else
    Response.Write "List failed."
End If

Set fc = Nothing
%>
```

328 Use the G3MD Library

328.1 Overview

The **G3MD** library provides high-performance Markdown-to-HTML conversion services for G3Pix AxonASP applications. Built on the industry-standard Goldmark engine, it supports GitHub Flavored Markdown (GFM) extensions, including tables, task lists, and strikethrough. The library is optimized for zero-allocation performance and is ideal for rendering technical documentation, blog posts, or user-generated content directly on the server.

328.2 Syntax

To instantiate the library, use the following syntax:

```
Set md = Server.CreateObject("G3MD")
```

328.3 Prerequisites

No external dependencies are required. The G3MD library is a built-in native component of the G3Pix AxonASP environment.

328.4 How it Works

The G3MD object operates as a stateful processor. You can configure conversion options such as **HardWraps** and **Unsafe** before calling the **Process** method. - **GitHub Flavored Markdown**: By default, the library uses the GFM extension set, ensuring compatibility with modern Markdown standards. - **Safety and Security**: The **Unsafe** property allows you to control whether raw HTML and potentially dangerous links are rendered, providing a layer of protection against cross-site scripting (XSS) when processing untrusted input. - **Line Break Handling**: The **HardWraps** property determines how soft line breaks in the Markdown source are translated into the final HTML output.

328.5 API Reference

328.5.1 Methods

- **Process**: Converts a Markdown string into a string of formatted HTML.

328.5.2 Properties

- **HardWraps**: Gets or sets a Boolean value indicating whether soft line breaks should be converted to `<br>` tags.
- **Unsafe**: Gets or sets a Boolean value indicating whether raw HTML and dangerous URLs should be rendered.

328.6 Code Example

The following example demonstrates how to configure the library and convert a Markdown string into HTML.

```
<%
Dim md, markdownText, htmlOutput
Set md = Server.CreateObject("G3MD")

' Configure the processor
md.HardWraps = True
md.Unsafe = False

' Markdown source with GFM features
markdownText = "# Welcome to AxonASP" & vbCrLf & _
               "This is a **high-performance** server." & vbCrLf & _
               "- Feature A" & vbCrLf & _
               "- Feature B"

' Convert to HTML
htmlOutput = md.Process(markdownText)

' Output the result
Response.Write htmlOutput
```

```
Set md = Nothing
%>
```

329 G3MD Methods

329.1 Overview

This page summarizes methods available in the G3MD library for Markdown processing in G3Pix AxonASP.

329.2 Methods Reference

Method	Returns	Description
Process	String	Converts Markdown source text to HTML output using the current object options.

329.3 Remarks

- Method names are case-insensitive.
- G3MD uses GitHub Flavored Markdown extensions.

330 G3MD Properties

330.1 Overview

This page summarizes properties available in the G3MD library for Markdown rendering configuration.

330.2 Properties Reference

Property	Access	Type	Description
HardWraps	Read/Write	Boolean	When True, converts soft line breaks into HTML line breaks.
Unsafe	Read/Write	Boolean	When True, allows raw HTML and non-sanitized links in rendered output.

330.3 Remarks

- Property names are case-insensitive.

- Property values affect subsequent Process calls on the same instance.

331 Process Method

331.1 Overview

Converts a Markdown-formatted string into a string of formatted HTML in the G3Pix AxonASP G3MD library.

331.2 Syntax

```
htmlString = md.Process(markdownText)
```

331.3 Parameters and Arguments

- **markdownText** (String, Required): The source text in Markdown format to be converted.

331.4 Return Values

Returns a **String** containing the rendered HTML. If the input is empty or an error occurs during conversion, it returns an empty string.

331.5 Remarks

- The output follows the GitHub Flavored Markdown (GFM) specification, including support for tables and task lists.
- The conversion process respects the current settings of the **HardWraps** and **Unsafe** properties.
- This method is highly optimized for performance and is suitable for high-traffic backend rendering.

331.6 Code Example

```
<%
Dim md, markdown, html
Set md = Server.CreateObject("G3MD")

markdown = "### System Status" & vbCrLf & "- **CPU**: 10%" & vbCrLf & "-
**Memory**: 2GB"
html = md.Process(markdown)

Response.Write html
Set md = Nothing
%>
```

332 HardWraps Property

332.1 Overview

Gets or sets a Boolean value that determines how the G3Pix AxonASP G3MD library handles soft line breaks in the Markdown source.

332.2 Syntax

```
' Get the current value
isHardWrap = md.HardWraps

' Set a new value
md.HardWraps = True
```

332.3 Return Values

Returns a **Boolean**. The default value is **False**.

332.4 Remarks

- When set to **True**, every soft line break (a single newline) in the Markdown source is converted to an HTML <br> tag.
- When set to **False** (default), Markdown follows standard behavior where soft line breaks are treated as spaces unless they are preceded by two spaces or followed by a blank line.
- This property must be set before calling the **Process** method.

332.5 Code Example

```
<%
Dim md, result
Set md = Server.CreateObject("G3MD")

' Enable hard wraps for comment-style rendering
md.HardWraps = True

result = md.Process("Line 1" & vbCrLf & "Line 2")
' Result will contain "Line 1<br>Line 2"

Response.Write result
Set md = Nothing
%>
```

333 Unsafe Property

333.1 Overview

Gets or sets a Boolean value that determines whether the G3Pix AxonASP G3MD library renders raw HTML and potentially dangerous URLs in the Markdown source.

333.2 Syntax

```
' Get the current value
isUnsafe = md.Unsafe

' Set a new value
md.Unsafe = True
```

333.3 Return Values

Returns a **Boolean**. The default value is **False**.

333.4 Remarks

- When set to **False** (default), the processor sanitizes the output, omitting raw HTML tags and dangerous link protocols (like `javascript:`) to protect against cross-site scripting (XSS).
- When set to **True**, the processor renders the Markdown exactly as provided, including any embedded HTML.
- Use **Unsafe = True** only when processing content from a trusted source or when your application requires specific HTML embedding.

333.5 Code Example

```
<%
Dim md, content, html
Set md = Server.CreateObject("G3MD")

' Securely render user-provided content
md.Unsafe = False

content = "Click [here](javascript:alert('XSS')) or <script>doBad()</scri
pt>"
html = md.Process(content)

' The dangerous script and link will be neutralized in the output
Response.Write html
Set md = Nothing
%>
```

334 Create PDF documents using G3PDF

334.1 Overview

The `G3PDF` object is a high-performance native library for generating PDF documents directly from Classic ASP code. It provides functions to add pages, format text, draw shapes, and manage document layout.

334.2 Syntax

```
Dim pdf
Set pdf = Server.CreateObject("G3PDF")
```

334.3 Parameters

None for instantiation.

334.4 Return Values

Returns a native `G3PDF` object that can be used to construct PDF documents.

334.5 Remarks

- Requires `Server.CreateObject("G3PDF")`. No aliases are supported.
- Powered by `go-pdf/fpdf` optimized for AxonASP.
- Ensure that memory and binary output streams are managed correctly.
- Call `Close` when finished applying content to the PDF object.

334.6 Code Example

```
<%
Dim pdf
Set pdf = Server.CreateObject("G3PDF")

pdf.AddPage "", "", 0
pdf.SetFont "Arial", "B", 16
pdf.Cell 40, 10, "Hello World!", 1, 0, "C", False, ""

' Further output code...
%>
```

335 G3PDF Methods

335.1 Overview

This page summarizes methods exposed by the G3PDF library in G3Pix AxonASP.

335.2 Methods Reference

Method	Returns	Description
New	Object	Initializes a new PDF context and returns the same object instance.
Init	Object	Alias of New. Initializes a new PDF context and returns the same object instance.
Reset	Object	Alias of New. Reinitializes the PDF context and returns the same object instance.
AddPage	Boolean	Adds a new page to the current document.
Close	Boolean	Finalizes the current document state.
Output	Boolean, String, or Null	Emits document output by destination mode: inline/download/file returns Boolean, string mode returns PDF binary string, and generation failures return Null.
SetFont	Boolean	Sets active font family, style, and size.
SetFontSize	Boolean	Sets active font size.
SetTextColor	Boolean	Sets text color using grayscale or RGB values.
SetDrawColor	Boolean	Sets stroke color using grayscale or RGB values.
SetFillColor	Boolean	Sets fill color using grayscale or RGB values.
SetLineWidth	Boolean	Sets line width for draw operations.
SetMargins	Boolean	Sets left, top, and optional right page margins. Returns False when

Method	Returns	Description
		required arguments are missing.
SetLeftMargin	Boolean	Sets left page margin.
SetTopMargin	Boolean	Sets top page margin.
SetRightMargin	Boolean	Sets right page margin.
SetX	Boolean	Sets the current horizontal cursor position.
SetY	Boolean	Sets the current vertical cursor position with optional X reset behavior.
SetXY	Boolean	Sets both horizontal and vertical cursor positions.
GetX	Double	Returns current horizontal cursor position.
GetY	Double	Returns current vertical cursor position.
Ln	Boolean	Moves cursor to the next line using optional offset.
Cell	Boolean	Writes one cell at the current cursor position. Returns False when required arguments are missing.
MultiCell	Boolean	Writes wrapped multi-line text cells. Returns False when required arguments are missing.
Write	Boolean	Writes flowing text. Returns False when required arguments are missing.
Text	Boolean	Writes text at absolute coordinates. Returns False when required arguments are missing.
Line	Boolean	Draws a line segment. Returns False when

Method	Returns	Description
		required arguments are missing.
Rect	Boolean	Draws a rectangle with optional style. Returns False when required arguments are missing.
Image	Boolean	Places an image at coordinates with optional sizing and link metadata. Returns False when required arguments are missing.
AddLink	Integer	Creates an internal link identifier and returns it.
SetLink	Boolean	Binds a link identifier to a document location. Returns False when required arguments are missing.
Link	Boolean	Creates a clickable rectangle bound to an internal or external link target. Returns False when required arguments are missing.
SetTitle	Boolean	Sets document title metadata.
SetAuthor	Boolean	Sets document author metadata.
SetSubject	Boolean	Sets document subject metadata.
SetKeywords	Boolean	Sets document keywords metadata.
SetCreator	Boolean	Sets document creator metadata.
AliasNbPages	Boolean	Sets the total-page alias token used in content placeholders.

Method	Returns	Description
SetDisplayMode	Boolean	Sets PDF viewer zoom and page layout mode.
SetCompression	Boolean	Enables or disables PDF stream compression.
WriteHTML	Boolean	Renders supported HTML markup into the current document. Returns False when HTML input is missing.
HTML	Boolean	Alias of WriteHTML.
WriteHTMLFile	Boolean	Loads and renders HTML from a file path. Returns False when the argument is missing or file loading fails.
HTMLFile	Boolean	Alias of WriteHTMLFile.
LoadHTMLFile	Boolean	Alias of WriteHTMLFile.
GetPageWidth	Double	Returns current page width in the active unit.
GetPageHeight	Double	Returns current page height in the active unit.
GetStringWidth	Double	Measures rendered width for a text string in the active font settings.

335.3 Remarks

- Method names are case-insensitive.
- Alias methods resolve to the same behavior and return contract as their canonical methods.

336 G3PDF Properties

336.1 Overview

This page summarizes properties exposed by the G3PDF library in G3Pix AxonASP.

336.2 Properties Reference

Property	Access	Type	Description

Property	Access	Type	Description
LastError	Read-only	String	Stores the most recent library error message.
Page	Read-only	Integer	Returns the current page number.
X	Read/Write	Double	Gets or sets the current horizontal cursor position.
Y	Read/Write	Double	Gets or sets the current vertical cursor position.
W	Read-only	Double	Returns current page width.
H	Read-only	Double	Returns current page height.
Version	Read-only	String	Returns the implementation version string for this library.

336.3 Remarks

- Property names are case-insensitive.
- PageWidth and PageHeight are accepted aliases for W and H.

337 AddPage

337.1 Overview

Adds a new page to the PDF document. Must be called before outputting any content to a page.

337.2 Syntax

```
obj.AddPage [orientation], [size]
```

337.3 Parameters

- `orientation` (String, Optional): Document orientation. Possible values are `P` (Portrait) or `L` (Landscape).
- `size` (String, Optional): Page format (e.g., `A4`, `Letter`).

337.4 Return Value

Returns: Empty

337.5 Code Example

```
<%
Option Explicit

Dim pdf
Set pdf = Server.CreateObject("G3PDF")

pdf.Reset "P", "mm", "A4"
pdf.AddPage

' Perform method operations here

Set pdf = Nothing
%>
```

338 Cell

338.1 Overview

Prints a cell (rectangular area) with optional borders, background color, and character string. The upper-left corner of the cell corresponds to the current position.

338.2 Syntax

```
obj.Cell w, h, txt, [border], [ln], [align], [fill], [link]
```

338.3 Parameters

- `w` (Double): Cell width. If 0, the cell extends up to the right margin.
- `h` (Double): Cell height.
- `txt` (String): String to print.
- `border` (Variant, Optional): Indicates if borders must be drawn (0, 1, or string like `LRTB`).
- `ln` (Integer, Optional): Indicates where the current position should go after the call (0: to the right, 1: to the beginning of the next line, 2: below).
- `align` (String, Optional): Text alignment (`L`, `C`, `R`).
- `fill` (Boolean, Optional): Indicates if the cell background must be painted.
- `link` (Integer, Optional): Identifier or string for a link.

338.4 Return Value

Returns: Empty

338.5 Code Example

```
<%
Option Explicit

Dim pdf
Set pdf = Server.CreateObject("G3PDF")

pdf.Reset "P", "mm", "A4"
pdf.AddPage

' Perform method operations here

Set pdf = Nothing
%>
```

339 Close

339.1 Overview

Terminates the PDF document. It is not necessary to call this method explicitly for most outputs as it is called automatically by output methods.

339.2 Syntax

```
obj.Close
```

339.3 Parameters

None.

339.4 Return Value

Returns: Empty

339.5 Code Example

```
<%
Option Explicit

Dim pdf
Set pdf = Server.CreateObject("G3PDF")

pdf.Reset "P", "mm", "A4"
pdf.AddPage

' Perform method operations here
```

```
Set pdf = Nothing
%>
```

340 Image

340.1 Overview

Outputs an image into the document.

340.2 Syntax

```
obj.Image imageName, x, y, [w], [h], [type], [link]
```

340.3 Parameters

- `imageName` (String): Path or URL of the image.
- `x` (Double): Abscissa of the upper-left corner.
- `y` (Double): Ordinate of the upper-left corner.
- `w` (Double, Optional): Width of the image in the page.
- `h` (Double, Optional): Height of the image in the page.
- `type` (String, Optional): Image format (e.g., `JPG`, `PNG`, `GIF`).
- `link` (Variant, Optional): URL or identifier.

340.4 Return Value

Returns: Empty

340.5 Code Example

```
<%
Option Explicit

Dim pdf
Set pdf = Server.CreateObject("G3PDF")

pdf.Reset "P", "mm", "A4"
pdf.AddPage

' Perform method operations here

Set pdf = Nothing
%>
```

341 Line

341.1 Overview

Draws a line between two points.

341.2 Syntax

```
obj.Line x1, y1, x2, y2
```

341.3 Parameters

- x1 (Double): Abscissa of first point.
- y1 (Double): Ordinate of first point.
- x2 (Double): Abscissa of second point.
- y2 (Double): Ordinate of second point.

341.4 Return Value

Returns: Empty

341.5 Code Example

```
<%
Option Explicit

Dim pdf
Set pdf = Server.CreateObject("G3PDF")

pdf.Reset "P", "mm", "A4"
pdf.AddPage

' Perform method operations here

Set pdf = Nothing
%>
```

342 Ln

342.1 Overview

Performs a line break. The current abscissa goes back to the left margin and the ordinate increases by the amount passed in parameter.

342.2 Syntax

```
obj.Ln [h]
```

342.3 Parameters

- h (Double, Optional): The height of the break. By default, the value equals the height of the last printed cell.

342.4 Return Value

Returns: Empty

342.5 Code Example

```
<%
Option Explicit

Dim pdf
Set pdf = Server.CreateObject("G3PDF")

pdf.Reset "P", "mm", "A4"
pdf.AddPage

' Perform method operations here

Set pdf = Nothing
%>
```

343 MultiCell

343.1 Overview

Allows printing text with line breaks. They can be automatic (as soon as the text reaches the right border of the cell) or explicit (via the newline character).

343.2 Syntax

```
obj.MultiCell w, h, txt, [border], [align], [fill]
```

343.3 Parameters

- `w` (Double): Width of cells. If 0, they extend up to the right margin of the page.
- `h` (Double): Height of cells.
- `txt` (String): String to print.
- `border` (Variant, Optional): Indicates if borders must be drawn (0, 1, or string like `LRTB`).
- `align` (String, Optional): Allows to center or align the text. Possible values: `L`, `C`, `R`, `J`.
- `fill` (Boolean, Optional): Indicates if the cell background must be painted.

343.4 Return Value

Returns: Empty

343.5 Code Example

```
<%
Option Explicit

Dim pdf
```

```
Set pdf = Server.CreateObject("G3PDF")

pdf.Reset "P", "mm", "A4"
pdf.AddPage

' Perform method operations here

Set pdf = Nothing
%>
```

344 Rect

344.1 Overview

Outputs a rectangle.

344.2 Syntax

```
obj.Rect x, y, w, h, [style]
```

344.3 Parameters

- `x` (Double): Abscissa of upper-left corner.
- `y` (Double): Ordinate of upper-left corner.
- `w` (Double): Width.
- `h` (Double): Height.
- `style` (String, Optional): Style of rendering. Possible values: `D` (Draw), `F` (Fill), `DF` or `FD` (Draw and fill).

344.4 Return Value

Returns: Empty

344.5 Code Example

```
<%
Option Explicit

Dim pdf
Set pdf = Server.CreateObject("G3PDF")

pdf.Reset "P", "mm", "A4"
pdf.AddPage

' Perform method operations here

Set pdf = Nothing
%>
```

345 Reset

345.1 Overview

Resets the PDF engine, clearing all content, settings, and pages. It optionally allows setting the new defaults for the document.

345.2 Syntax

```
obj.Reset [orientation], [unit], [size]
```

345.3 Parameters

- `orientation` (String, Optional): Default page orientation (`P` or `L`).
- `unit` (String, Optional): User unit of measure (e.g., `pt`, `mm`, `cm`, `in`).
- `size` (String, Optional): Default page format (e.g., `A4`, `Letter`).

345.4 Return Value

Returns: Empty

345.5 Code Example

```
<%
Option Explicit

Dim pdf
Set pdf = Server.CreateObject("G3PDF")

pdf.Reset "P", "mm", "A4"
pdf.AddPage

' Perform method operations here

Set pdf = Nothing
%>
```

346 SetAutoPageBreak

346.1 Overview

Enables or disables the automatic page breaking mode.

346.2 Syntax

```
obj.SetAutoPageBreak auto, [margin]
```

346.3 Parameters

- `auto` (Boolean): Boolean indicating if mode should be on or off.

- `margin` (Double, Optional): Distance from the bottom of the page that triggers the break.

346.4 Return Value

Returns: Empty

346.5 Code Example

```
<%
Option Explicit

Dim pdf
Set pdf = Server.CreateObject("G3PDF")

pdf.Reset "P", "mm", "A4"
pdf.AddPage

' Perform method operations here

Set pdf = Nothing
%>
```

347 SetDrawColor

347.1 Overview

Defines the color used for all drawing operations (lines, rectangles and cell borders). It can be expressed in RGB components or grayscale.

347.2 Syntax

```
obj.SetDrawColor r, [g], [b]
```

347.3 Parameters

- `r` (Integer): If `g` and `b` are given, this indicates the red component; else, it indicates the grayscale level.
- `g` (Integer, Optional): Green component (between 0 and 255).
- `b` (Integer, Optional): Blue component (between 0 and 255).

347.4 Return Value

Returns: Empty

347.5 Code Example

```
<%
Option Explicit
```

```
Dim pdf
Set pdf = Server.CreateObject("G3PDF")

pdf.Reset "P", "mm", "A4"
pdf.AddPage

' Perform method operations here

Set pdf = Nothing
%>
```

348 SetFillColor

348.1 Overview

Defines the color used for all filling operations (filled rectangles and cell backgrounds).

348.2 Syntax

```
obj.SetFillColor r, [g], [b]
```

348.3 Parameters

- `r` (Integer): Red component or grayscale level.
- `g` (Integer, Optional): Green component.
- `b` (Integer, Optional): Blue component.

348.4 Return Value

Returns: Empty

348.5 Code Example

```
<%
Option Explicit

Dim pdf
Set pdf = Server.CreateObject("G3PDF")

pdf.Reset "P", "mm", "A4"
pdf.AddPage

' Perform method operations here

Set pdf = Nothing
%>
```

349 SetFont

349.1 Overview

Sets the font used to print character strings.

349.2 Syntax

```
obj.SetFont family, [style], [size]
```

349.3 Parameters

- `family` (String): Family font. It can be a standard TrueType/Type1 font.
- `style` (String, Optional): Font style. Possible values are empty string (regular), `B` (bold), `I` (italic), `U` (underline), or any combination.
- `size` (Double, Optional): Font size in points.

349.4 Return Value

Returns: Empty

349.5 Code Example

```
<%
Option Explicit

Dim pdf
Set pdf = Server.CreateObject("G3PDF")

pdf.Reset "P", "mm", "A4"
pdf.AddPage

' Perform method operations here

Set pdf = Nothing
%>
```

350 SetFontSize

350.1 Overview

Defines the size of the current font.

350.2 Syntax

```
obj.SetFontSize size
```

350.3 Parameters

- `size` (Double): The size (in points).

350.4 Return Value

Returns: Empty

350.5 Code Example

```
<%
Option Explicit

Dim pdf
Set pdf = Server.CreateObject("G3PDF")

pdf.Reset "P", "mm", "A4"
pdf.AddPage

' Perform method operations here

Set pdf = Nothing
%>
```

351 SetLeftMargin

351.1 Overview

Defines the left margin.

351.2 Syntax

```
obj.SetLeftMargin margin
```

351.3 Parameters

- `margin` (Double): The margin.

351.4 Return Value

Returns: Empty

351.5 Code Example

```
<%
Option Explicit

Dim pdf
Set pdf = Server.CreateObject("G3PDF")

pdf.Reset "P", "mm", "A4"
pdf.AddPage

' Perform method operations here
```

```
Set pdf = Nothing
%>
```

352 SetLineWidth

352.1 Overview

Defines the line width.

352.2 Syntax

```
obj.SetLineWidth width
```

352.3 Parameters

- `width` (Double): The width.

352.4 Return Value

Returns: Empty

352.5 Code Example

```
<%
Option Explicit

Dim pdf
Set pdf = Server.CreateObject("G3PDF")

pdf.Reset "P", "mm", "A4"
pdf.AddPage

' Perform method operations here

Set pdf = Nothing
%>
```

353 SetMargins

353.1 Overview

Defines the left, top, and right margins. By default, they equal 1 cm.

353.2 Syntax

```
obj.SetMargins left, top, [right]
```

353.3 Parameters

- `left` (Double): Left margin.

- `top` (Double): Top margin.
- `right` (Double, Optional): Right margin. Default to the value of the left margin.

353.4 Return Value

Returns: Empty

353.5 Code Example

```
<%
Option Explicit

Dim pdf
Set pdf = Server.CreateObject("G3PDF")

pdf.Reset "P", "mm", "A4"
pdf.AddPage

' Perform method operations here

Set pdf = Nothing
%>
```

354 SetRightMargin

354.1 Overview

Defines the right margin.

354.2 Syntax

```
obj.SetRightMargin margin
```

354.3 Parameters

- `margin` (Double): The margin.

354.4 Return Value

Returns: Empty

354.5 Code Example

```
<%
Option Explicit

Dim pdf
Set pdf = Server.CreateObject("G3PDF")

pdf.Reset "P", "mm", "A4"
```

```
pdf.AddPage

' Perform method operations here

Set pdf = Nothing
%>
```

355 SetTextColor

355.1 Overview

Defines the color used for text. It can be expressed in RGB components or grayscale.

355.2 Syntax

```
obj.SetTextColor r, [g], [b]
```

355.3 Parameters

- r (Integer): Red component or grayscale level.
- g (Integer, Optional): Green component.
- b (Integer, Optional): Blue component.

355.4 Return Value

Returns: Empty

355.5 Code Example

```
<%
Option Explicit

Dim pdf
Set pdf = Server.CreateObject("G3PDF")

pdf.Reset "P", "mm", "A4"
pdf.AddPage

' Perform method operations here

Set pdf = Nothing
%>
```

356 SetTopMargin

356.1 Overview

Defines the top margin.

356.2 Syntax

```
obj.SetTopMargin margin
```

356.3 Parameters

- `margin` (Double): The margin.

356.4 Return Value

Returns: Empty

356.5 Code Example

```
<%
Option Explicit

Dim pdf
Set pdf = Server.CreateObject("G3PDF")

pdf.Reset "P", "mm", "A4"
pdf.AddPage

' Perform method operations here

Set pdf = Nothing
%>
```

357 SetX

357.1 Overview

Defines the abscissa of the current position.

357.2 Syntax

```
obj.SetX x
```

357.3 Parameters

- `x` (Double): The value of the abscissa.

357.4 Return Value

Returns: Empty

357.5 Code Example

```
<%
Option Explicit

Dim pdf
```

```
Set pdf = Server.CreateObject("G3PDF")

pdf.Reset "P", "mm", "A4"
pdf.AddPage

' Perform method operations here

Set pdf = Nothing
%>
```

358 SetXY

358.1 Overview

Defines the abscissa and ordinate of the current position.

358.2 Syntax

```
obj.SetXY x, y
```

358.3 Parameters

- x (Double): The value of the abscissa.
- y (Double): The value of the ordinate.

358.4 Return Value

Returns: Empty

358.5 Code Example

```
<%
Option Explicit

Dim pdf
Set pdf = Server.CreateObject("G3PDF")

pdf.Reset "P", "mm", "A4"
pdf.AddPage

' Perform method operations here

Set pdf = Nothing
%>
```

359 SetY

359.1 Overview

Moves the current abscissa back to the left margin and sets the ordinate.

359.2 Syntax

```
obj.SetY y, [resetX]
```

359.3 Parameters

- y (Double): The value of the ordinate.
- resetX (Boolean, Optional): Indicates if the abscissa should be reset to the left margin. Default is true.

359.4 Return Value

Returns: Empty

359.5 Code Example

```
<%
Option Explicit

Dim pdf
Set pdf = Server.CreateObject("G3PDF")

pdf.Reset "P", "mm", "A4"
pdf.AddPage

' Perform method operations here

Set pdf = Nothing
%>
```

360 Text

360.1 Overview

Prints a character string. The origin is on the left of the first character, on the baseline.

360.2 Syntax

```
obj.Text x, y, txt
```

360.3 Parameters

- x (Double): Abscissa of the origin.
- y (Double): Ordinate of the origin.

- `txt` (String): String to print.

360.4 Return Value

Returns: Empty

360.5 Code Example

```
<%
Option Explicit

Dim pdf
Set pdf = Server.CreateObject("G3PDF")

pdf.Reset "P", "mm", "A4"
pdf.AddPage

' Perform method operations here

Set pdf = Nothing
%>
```

361 Write

361.1 Overview

Prints text from the current position. When the right margin is reached (or the newline character is met), a line break occurs and text continues from the left margin.

361.2 Syntax

```
obj.Write h, txt, [link]
```

361.3 Parameters

- `h` (Double): Line height.
- `txt` (String): String to print.
- `link` (Variant, Optional): URL or identifier.

361.4 Return Value

Returns: Empty

361.5 Code Example

```
<%
Option Explicit

Dim pdf
Set pdf = Server.CreateObject("G3PDF")
```

```
pdf.Reset "P", "mm", "A4"
pdf.AddPage

' Perform method operations here

Set pdf = Nothing
%>
```

362 H

362.1 Overview

Gets the current page height.

362.2 Property Type

Returns: Double

362.3 Code Example

```
<%
Option Explicit

Dim pdf, val
Set pdf = Server.CreateObject("G3PDF")

pdf.Reset "P", "mm", "A4"
pdf.AddPage

val = pdf.H
Response.Write CStr(val)

Set pdf = Nothing
%>
```

363 LastError

363.1 Overview

Returns the description of the last error encountered during operations. Returns an empty string if no error occurred.

363.2 Property Type

Returns: String

363.3 Code Example

```
<%
Option Explicit

Dim pdf, val
Set pdf = Server.CreateObject("G3PDF")

pdf.Reset "P", "mm", "A4"
pdf.AddPage

val = pdf.LastError
Response.Write CStr(val)

Set pdf = Nothing
%>
```

364 Page

364.1 Overview

Returns the current page number.

364.2 Property Type

Returns: Integer

364.3 Code Example

```
<%
Option Explicit

Dim pdf, val
Set pdf = Server.CreateObject("G3PDF")

pdf.Reset "P", "mm", "A4"
pdf.AddPage

val = pdf.Page
Response.Write CStr(val)

Set pdf = Nothing
%>
```

365 Version

365.1 Overview

Returns the current version string of the underlying go-pdf/fpdf library wrapper.

365.2 Property Type

Returns: String

365.3 Code Example

```
<%
Option Explicit

Dim pdf, val
Set pdf = Server.CreateObject("G3PDF")

pdf.Reset "P", "mm", "A4"
pdf.AddPage

val = pdf.Version
Response.Write CStr(val)

Set pdf = Nothing
%>
```

366 W

366.1 Overview

Gets the current page width.

366.2 Property Type

Returns: Double

366.3 Code Example

```
<%
Option Explicit

Dim pdf, val
Set pdf = Server.CreateObject("G3PDF")

pdf.Reset "P", "mm", "A4"
pdf.AddPage

val = pdf.W
Response.Write CStr(val)

Set pdf = Nothing
%>
```

367 X

367.1 Overview

Gets the abscissa of the current position.

367.2 Property Type

Returns: Double

367.3 Code Example

```
<%
Option Explicit

Dim pdf, val
Set pdf = Server.CreateObject("G3PDF")

pdf.Reset "P", "mm", "A4"
pdf.AddPage

val = pdf.X
Response.Write CStr(val)

Set pdf = Nothing
%>
```

368 Y

368.1 Overview

Gets the ordinate of the current position.

368.2 Property Type

Returns: Double

368.3 Code Example

```
<%
Option Explicit

Dim pdf, val
Set pdf = Server.CreateObject("G3PDF")

pdf.Reset "P", "mm", "A4"
pdf.AddPage

val = pdf.Y
```

```
Response.Write CStr(val)

Set pdf = Nothing
%>
```

369 G3FILEUPLOADER Library

369.1 Overview

The G3FILEUPLOADER object provides a high-performance system for securely processing and managing HTTP multipart file uploads in the AxonASP environment. You can control validation logic, restrict extensions, limit file sizes, and retrieve deep metadata describing individual files or the entire upload batch.

369.2 Syntax

```
Set uploader = Server.CreateObject("G3FILEUPLOADER")
```

369.3 Properties and Methods

The object exposes specific methods to parse forms and properties to refine the validation logic. View the properties and methods listing pages for details.

369.4 Return Values

Returns a native AxonASP object handle referencing the file uploader instance.

369.5 Example

```
<%
Dim uploader
Set uploader = Server.CreateObject("G3FILEUPLOADER")
uploader.MaxFileSize = 5242880 ' 5 MB
uploader.BlockExtension "exe"

Dim result
Set result = uploader.Process("myFile", "/uploads/docs", "newdocument.pdf
")
If result("IsSuccess") Then
    Response.Write "File saved at: " & result("RelativePath")
Else
    Response.Write "Error: " & result("ErrorMessage")
End If
%>
```

370 G3FILEUPLOADER Methods

370.1 Overview

This page summarizes methods exposed by G3FILEUPLOADER in G3Pix AxonASP for upload validation, processing, and metadata inspection.

370.2 Methods Reference

Method	Returns	Description
BlockExtension	Empty	Adds one extension to the blocked list.
AllowExtension	Empty	Adds one extension to the allowed list.
BlockExtensions	Empty	Adds multiple extensions to the blocked list from a comma-separated string.
AllowExtensions	Empty	Adds multiple extensions to the allowed list from a comma-separated string.
SetUseAllowedOnly	Empty	Enables or disables allow-list-only validation mode.
Process	Dictionary or Empty	Processes one uploaded file and returns a dictionary with success flag, names, size, MIME type, and error message. Returns Empty when required arguments are missing.
Save	Dictionary or Empty	Alias of Process.
ProcessAll	Array of Dictionary	Processes all uploaded files and returns one result dictionary per file.
SaveAll	Array of Dictionary	Alias of ProcessAll.
GetFileInfo	Dictionary or Empty	Returns metadata for one uploaded form field. Returns Empty when request context is unavailable.

Method	Returns	Description
GetAllFilesInfo	Array of Dictionary	Returns metadata for all uploaded files in the current multipart request.
Form	String or Empty	Gets a multipart form field value by name. Returns Empty when not found.
FormValue	String or Empty	Alias of Form.
IsValidExtension	Boolean	Validates whether an extension is currently allowed under configured rules.

370.3 Remarks

- Method names are case-insensitive.
- Result dictionaries include IsSuccess and ErrorMessage keys for operation status.

371 G3FILEUPLOADER Properties

371.1 Overview

This page summarizes properties exposed by G3FILEUPLOADER in G3Pix AxonASP.

371.2 Properties Reference

Property	Access	Type	Description
BlockedExtensions	Read-only	Array of String	Gets the current blocked extension set.
AllowedExtensions	Read-only	Array of String	Gets the current allowed extension set.
MaxFileSize	Read/Write	Integer	Gets or sets the maximum upload size in bytes for each file.
PreserveOriginalName	Read/Write	Boolean	Gets or sets whether saved files preserve the original uploaded

Property	Access	Type	Description
			filename.
DebugMode	Read/Write	Boolean	Gets or sets debug mode for uploader behavior.

371.3 Remarks

- Property names are case-insensitive.
- Use AllowExtension, AllowExtensions, BlockExtension, and BlockExtensions to mutate extension lists.

372 AllowExtension Method

372.1 Overview

Appends a single file extension to the internally maintained allowed extensions permit list. Usually used in tandem with the `SetUseAllowedOnly` mechanism.

372.2 Syntax

```
Set uploader = Server.CreateObject("G3FILEUPLOADER")
uploader.AllowExtension ".pdf"
```

372.3 Parameters and Arguments

- `Extension` (String, Required): The literal file extension (with or without the leading dot).

372.4 Return Values

Returns an `Empty` variant.

373 AllowExtensions Method

373.1 Overview

Bulk injects multiple allowed file extensions by processing a comma-delimited string of formats.

373.2 Syntax

```
Set uploader = Server.CreateObject("G3FILEUPLOADER")
uploader.AllowExtensions "pdf,doc,docx"
```

373.3 Parameters and Arguments

- `ExtensionsList` (String, Required): A comma-separated list of file extensions.

373.4 Return Values

Returns an `Empty` variant.

374 BlockExtension Method

374.1 Overview

Registers a specific file extension to heavily restrict it. The system automatically rejects uploads containing this extension.

374.2 Syntax

```
Set uploader = Server.CreateObject("G3FILEUPLOADER")
uploader.BlockExtension "php"
```

374.3 Parameters and Arguments

- `Extension` (String, Required): The prohibited file format extension.

374.4 Return Values

Returns an `Empty` variant.

375 BlockExtensions Method

375.1 Overview

Submits a comma-separated format string to define multiple blocked file extensions simultaneously.

375.2 Syntax

```
Set uploader = Server.CreateObject("G3FILEUPLOADER")
uploader.BlockExtensions "jsp,asp,exe,dll,bat"
```

375.3 Parameters and Arguments

- `ExtensionsList` (String, Required): A robust comma-separated blocklist of targeted formats.

375.4 Return Values

Returns an `Empty` variant.

376 Form Method

376.1 Overview

Equivalent to `Request.Form("key")` but safely integrated with multipart form data upload flows. Given that normal request processing may be trapped by stream bodies, this extracts standard form field items alongside file uploads. The alias `FormValue` is also supported.

376.2 Syntax

```
Set uploader = Server.CreateObject("G3FILEUPLOADER")
Dim description
description = uploader.Form("fileDescription")
```

376.3 Parameters and Arguments

- `FieldName` (String, Required): The HTML element `name` variable of the standard form field.

376.4 Return Values

Returns a string representing the extracted value, or `Empty` if not found or if form analysis fails completely.

377 GetAllFilesInfo Method

377.1 Overview

Performs an architectural scan against the intercepted HTTP payload, surfacing comprehensive data dictionaries representing every uploaded item, bypassing persistence entirely.

377.2 Syntax

```
Set uploader = Server.CreateObject("G3FILEUPLOADER")
Dim fileArray
fileArray = uploader.GetAllFilesInfo()
```

377.3 Return Values

Returns a VBScript array filled with Dictionary objects. Each Dictionary contains attributes analogous to `GetFileInfo`. If no multipart form data exists, returns an empty VBScript array.

378 GetFileInfo Method

378.1 Overview

Extracts data related to a specified inbound form file input before you commit the asset onto system storage via the primary methods.

378.2 Syntax

```
Set uploader = Server.CreateObject("G3FILEUPLOADER")
Dim details
Set details = uploader.GetFileInfo("document")
```

378.3 Parameters and Arguments

- `FieldName` (String, Required): The exact string moniker denoting the form input parameter mapping to a physical attachment stream.

378.4 Return Values

Returns a Dictionary object populated mapping string configurations, or a Dictionary representing failure (with `IsSuccess` set to false) if missing details occur, or `Empty` if no HTTP request context is completely available.

379 IsValidExtension Method

379.1 Overview

Tests logic sequentially running validation over blocked arrays, max limits, against any strictly targeted file type extension, checking engine integrity dynamically without causing data persistence failures directly handling bytes.

379.2 Syntax

```
Set uploader = Server.CreateObject("G3FILEUPLOADER")
If uploader.IsValidExtension(".xlsx") Then
    ' Valid configuration
End If
```

379.3 Parameters and Arguments

- `ExtensionName` (String, Required): A string file type identifier (e.g., “.txt”).

379.4 Return Values

Returns a boolean representing true if the file validates perfectly under existing internal limits.

380 Process Method

380.1 Overview

Processes a specific attached form file payload byte stream, pushing it directly to an allocated destination directory on disk, optionally allowing overrides of the core internal naming generator. Also supports the `Save` alias.

380.2 Syntax

```
Set uploader = Server.CreateObject("G3FILEUPLOADER")
Dim res
Set res = uploader.Process("myFile", "/uploads/", "forcedName.png")
```

380.3 Parameters and Arguments

- `FieldName` (String, Required): File form field identifier.
- `TargetDir` (String, Optional): Destination virtual directory to save the file (Defaults to "./").
- `NewFileName` (String, Optional): Explicit string name formatting option ignoring internal hash behaviors.

380.4 Return Values

Returns a Dictionary object structured heavily reflecting successful operation indicators (via a boolean `IsSuccess` property). Other Dictionary keys include `OriginalFileName`, `NewFileName`, `Size`, `MimeType`, `Extension`, `FinalPath`, `RelativePath`, and `ErrorMessage`.

381 ProcessAll Method

381.1 Overview

Saves all pending multipart files embedded directly onto the stream request simultaneously parsing the system configuration. Validates limits, unique variables, and dynamically writes to the filesystem tracking the final status for each target. Also accepts the `SaveAll` alias.

381.2 Syntax

```
Set uploader = Server.CreateObject("G3FILEUPLOADER")
Dim resultsList
resultsList = uploader.ProcessAll("/uploads/")
```

381.3 Parameters and Arguments

- `TargetDir` (String, Optional): Destination virtual directory to securely organize distribution of all incoming items (Defaults to "./").

381.4 Return Values

Returns a VBScript array containing separate Dictionary objects mapping independent results for each item processed (sharing attributes akin to `Process`), such as `IsSuccess` and related naming metrics to capture standard logs precisely.

382 SetUseAllowedOnly Method

382.1 Overview

Activates an internal strict lockdown, discarding all system capabilities globally barring those extensions exclusively pushed directly into engine array mechanisms matching precise whitelist entries.

382.2 Syntax

```
Set uploader = Server.CreateObject("G3FILEUPLOADER")
uploader.SetUseAllowedOnly True
```

382.3 Parameters and Arguments

- `Enabled` (Boolean, Required): True directly configures whitelist-only uploads.

382.4 Return Values

Returns an `Empty` variant.

383 AllowedExtensions Property

383.1 Overview

A readonly property that returns an array of explicitly permitted file extensions currently loaded into memory.

383.2 Syntax

```
Set uploader = Server.CreateObject("G3FILEUPLOADER")
Dim list
list = uploader.AllowedExtensions
```

383.3 Return Values

Returns a VBScript array containing strings of all allowed file extensions.

384 BlockedExtensions Property

384.1 Overview

A readonly property that returns an array of explicitly forbidden file extensions loaded into the validation engine.

384.2 Syntax

```
Set uploader = Server.CreateObject("G3FILEUPLOADER")
Dim list
list = uploader.BlockedExtensions
```

384.3 Return Values

Returns a VBScript array containing strings of all strictly blocked file extensions.

385 DebugMode Property

385.1 Overview

Configures whether the file uploader should run with debug-level internal tracking. By default, this is disabled.

385.2 Syntax

```
Set uploader = Server.CreateObject("G3FILEUPLOADER")
uploader.DebugMode = True
```

385.3 Return Values

Sets or returns a boolean indicating whether debug mode is enabled.

386 MaxFileSize Property

386.1 Overview

Controls the maximum permitted file size (in bytes) per individual request. By default, this is set to 10 MB (10485760 bytes).

386.2 Syntax

```
Set uploader = Server.CreateObject("G3FILEUPLOADER")
uploader.MaxFileSize = 5242880 ' 5 MB maximum
```

386.3 Return Values

Sets or returns a long integer reflecting the maximum file size configured in the uploader instance.

387 PreserveOriginalName Property

387.1 Overview

Instructs the uploader engine to bypass its internal random hash generator and save files using their exact original names as submitted by the client device.

387.2 Syntax

```
Set uploader = Server.CreateObject("G3FILEUPLOADER")
uploader.PreserveOriginalName = True
```

387.3 Return Values

Sets or returns a boolean reflecting the name preservation rule.

388 Use WScript.Shell in AxonASP

388.1 Overview

Process and environment shell object.

388.2 Syntax

```
Set obj = Server.CreateObject("WScript.Shell")
```

388.3 Parameters and Arguments

- ProgID (String, Required): Use one of the supported ProgID forms for this object family.
- Member access (Optional): Use documented method/property members from the library reference pages.

388.4 Return Values

Returns a native object handle for this object family.

388.5 Remarks

- Member names are case-insensitive.
- Runtime validation is enforced by object dispatch logic.
- See the central library methods/properties pages for member-level coverage.

388.6 Code Example

```
<%
Dim obj
Set obj = Server.CreateObject("WScript.Shell")
Response.Write TypeName(obj)
%>
```

389 WScript.Shell Methods

389.1 Overview

This page summarizes core methods exposed by the WScript.Shell library in G3Pix AxonASP.

389.2 Methods Reference

Method	Returns	Description
Run(command [, windowStyle] [, waitOnReturn])	Integer	Executes a command line. Returns 0 on successful start and successful completion, non-zero process exit code when waitOnReturn is True and the process fails, or -1 when command is missing, empty, or startup fails.
Exec(command)	Object or Empty	Starts a process asynchronously and returns a process object with stream and status members. Returns Empty when command is missing, empty, or startup fails.
CreateObject(progID)	Object or Empty	Creates an object using the AxonASP object factory. Returns an object reference on success, or Empty when progID is missing, empty, or object creation fails.
GetEnv(name)	String	Returns the value of one environment variable. Returns an empty string when name is missing or the variable does not exist.
ExpandEnvironmentString	String	Expands percent-delimited

Method	Returns	Description
s(inputText)		environment placeholders such as %PATH%. Returns an empty string when inputText is missing. Unknown placeholders are preserved.

389.3 Remarks

- Method names are case-insensitive.
- Use Set for object-returning methods such as Exec and CreateObject.

390 WScript.Shell Properties

390.1 Overview

This page summarizes properties exposed by the WScript.Shell library in G3Pix AxonASP.

390.2 Properties Reference

Property	Access	Type	Description
None	Not applicable	Not applicable	WScript.Shell does not expose dedicated library properties in this runtime. Property-style access is routed to method dispatch.

390.3 Remarks

- Method-style members provide the supported runtime behavior for this library.

391 Use CreateObject from WScript.Shell

391.1 Overview

Use CreateObject to instantiate another COM-style object through the same runtime path used by Server.CreateObject.

391.2 Syntax

```
Set result = shell.CreateObject(progID)
```

391.3 Parameters

- progID (String, required): Programmatic identifier of the object to create, such as G3JSON, Scripting.Dictionary, or WScript.Shell.

391.4 Return Value

Returns an object reference when creation succeeds.

Returns Empty when progID is missing, blank, or object creation fails.

391.5 How It Works

- WScript.Shell.CreateObject delegates to the same backend object factory used by Server.CreateObject.
- The method does not return a Boolean status. Check the returned value with IsObject.

391.6 Remarks

- Method names are case-insensitive.
- Use Set when assigning the returned object reference.

391.7 Example

```
<%
Option Explicit

Dim shell, dict
Set shell = Server.CreateObject("WScript.Shell")
Set dict = shell.CreateObject("Scripting.Dictionary")

If IsObject(dict) Then
    dict.Add "engine", "G3Pix AxonASP"
    Response.Write dict.Item("engine")
Else
    Response.Write "CreateObject failed"
End If

Set dict = Nothing
Set shell = Nothing
%>
```

392 Run a Command with Exec

392.1 Overview

Use Exec to start a process and interact with it through a process object that exposes output streams and status information.

392.2 Syntax

```
Set execObj = shell.Exec(command)
```

392.3 Parameters

- command (String, required): Command line to execute.

392.4 Return Value

Returns an object reference that represents the running process when process startup succeeds.

Returns Empty when command is missing, blank, or process startup fails.

392.5 How It Works

- On Windows, AxonASP runs the command through cmd.exe /c.
- The returned process object exposes members such as StdOut, StdErr, Status, ExitCode, and ProcessID.

392.6 Remarks

- Method names are case-insensitive.
- Use Set when assigning the returned process object.
- Exec starts asynchronously. Use process state members to track completion.

392.7 Example

```
<%
Option Explicit

Dim shell, execObj
Set shell = Server.CreateObject("WScript.Shell")
Set execObj = shell.Exec("echo AxonASP")

If IsObject(execObj) Then
    execObj.WaitUntilDone 5000
    Response.Write "ExitCode=" & execObj.ExitCode
Else
    Response.Write "Exec failed"
End If

Set execObj = Nothing
Set shell = Nothing
%>
```

393 Expand Environment Variables in Text

393.1 Overview

Use ExpandEnvironmentStrings to replace tokens in the format %NAME% with values from the current process environment.

393.2 Syntax

```
expanded = shell.ExpandEnvironmentStrings(inputText)
```

393.3 Parameters

- inputText (String, required): Text that may contain one or more %NAME% placeholders.

393.4 Return Value

Returns a String containing the expanded text.

Returns an empty string when inputText is missing.

393.5 How It Works

- Each %NAME% token is resolved using the current environment.
- Unknown variables are preserved as written, for example %UNKNOWN_VAR% remains unchanged.

393.6 Remarks

- Method names are case-insensitive.
- Expansion follows percent-delimited placeholder format only.

393.7 Example

```
<%
Option Explicit

Dim shell, expanded
Set shell = Server.CreateObject("WScript.Shell")

expanded = shell.ExpandEnvironmentStrings("PATH=%PATH%")
Response.Write expanded

Set shell = Nothing
%>
```

394 Read an Environment Variable with GetEnv

394.1 Overview

Use GetEnv to read one environment variable from the current process environment.

394.2 Syntax

```
value = shell.GetEnv(name)
```

394.3 Parameters

- name (String, required): Environment variable name, such as PATH, TEMP, or USERNAME.

394.4 Return Value

Returns a String containing the variable value when the variable exists.

Returns an empty string when name is missing or when the variable is not defined.

394.5 How It Works

- The method reads directly from the OS process environment seen by AxonASP.
- Missing variables do not raise an error. The method returns an empty string.

394.6 Remarks

- Method names are case-insensitive.
- The alias EnvironmentVariables is also accepted.

394.7 Example

```
<%
Option Explicit

Dim shell, pathValue
Set shell = Server.CreateObject("WScript.Shell")

pathValue = shell.GetEnv("PATH")
If pathValue = "" Then
    Response.Write "PATH is not available"
Else
    Response.Write "PATH length: " & Len(pathValue)
End If

Set shell = Nothing
%>
```

395 Run a Command with Run

395.1 Overview

Use Run to execute a command line and receive an Integer status code.

395.2 Syntax

```
exitCode = shell.Run(command, windowStyle, waitOnReturn)
```

395.3 Parameters

- command (String, required): Command line to execute.
- windowStyle (Integer, optional): Accepted for compatibility. Current AxonASP runtime does not apply this value.
- waitOnReturn (Boolean, optional): When True, waits for process completion before returning. Default is True.

395.4 Return Value

Returns Integer 0 when command start succeeds and either: - the process finishes successfully when waitOnReturn is True, or - the process starts successfully when waitOnReturn is False.

Returns an Integer greater than 0 when waitOnReturn is True and the process exits with a non-zero exit code.

Returns Integer -1 when command is missing, blank, or process startup fails.

395.5 How It Works

- On Windows, AxonASP executes the command through cmd.exe /c.
- On non-Windows environments, AxonASP executes through sh -c.

395.6 Remarks

- Method names are case-insensitive.
- Run does not return process output. Use Exec when you need StdOut or StdErr streams.

395.7 Example

```
<%
Option Explicit

Dim shell, code
Set shell = Server.CreateObject("WScript.Shell")

code = shell.Run("cmd /c echo G3Pix AxonASP", 0, True)
```

```
Response.Write "Run returned: " & code

Set shell = Nothing
%>
```

396 Use ADOX.Catalog in G3Pix AxonASP

396.1 Overview

Use ADOX.Catalog to inspect database table metadata through the catalog surface implemented by G3Pix AxonASP.

396.2 Prerequisites

- A valid ADO connection object or connection string.
- A database provider supported by the current runtime path.

396.3 Syntax

```
Set catalog = Server.CreateObject("ADOX.Catalog")
catalog.ActiveConnection = connectionObject
Set tables = catalog.Tables
```

396.4 Parameters and Arguments

- ProgID (String, required): Use ADOX.Catalog.
- ActiveConnection assignment value (required before reading schema):
  - ADODB connection object, or
  - connection string.

396.5 Return Value

`Server.CreateObject("ADOX.Catalog")` returns an object reference to the catalog instance.

396.6 How It Works

- The catalog stores ActiveConnection and resolves table metadata when Tables is read.
- On Windows, the runtime first attempts OLE schema discovery.
- When OLE is not available, the runtime uses native schema queries supported by the current connection backend.

396.7 Remarks

- Member names are case-insensitive.
- Tables is evaluated lazily and cached until ActiveConnection is changed.

396.8Example

```
<%
Option Explicit

Dim conn, catalog, tables
Set conn = Server.CreateObject("ADODB.Connection")
conn.Open "Data Source=./temp/sample.db;Version=3;"

Set catalog = Server.CreateObject("ADOX.Catalog")
Set catalog.ActiveConnection = conn
Set tables = catalog.Tables

Response.Write "Table count: " & tables.Count

Set tables = Nothing
Set catalog = Nothing
conn.Close
Set conn = Nothing
%>
```

397 ADOX.Catalog Methods

397.1 Overview

This page summarizes callable methods exposed directly by ADOX.Catalog in G3Pix AxonASP.

397.2 Methods Reference

Method	Returns	Description
None	Not applicable	The catalog object does not expose direct methods. Use properties such as ActiveConnection and Tables.

397.3 Remarks

- Method-style calls on the catalog object return Empty in this runtime.

398 ADOX.Catalog Properties

398.1 Overview

This page summarizes properties exposed by ADOX.Catalog in G3Pix AxonASP.

398.2 Properties Reference

Property	Access	Type	Description
ActiveConnection	Read/Write	Object or String or Empty	Stores the current catalog connection source. Assign an ADODB connection object or a connection string.
Tables	Read-only	Object	Returns an ADOX table collection object representing discovered tables and views for ActiveConnection.

398.3 Remarks

- Property names are case-insensitive.
- Assigning ActiveConnection resets cached Tables metadata.

399 Set Catalog ActiveConnection

399.1 Overview

Use ActiveConnection to define the connection source used by ADOX.Catalog for schema discovery.

399.2 Syntax

```
Set catalog.ActiveConnection = connectionObject
catalog.ActiveConnection = connectionString
value = catalog.ActiveConnection
```

399.3 Parameters

- Setter value (required for assignment):
  - ADODB connection object, or
  - connection string.

399.4 Return Value

Getter returns the value currently stored in ActiveConnection: - Object when set with an ADODB connection object. - String when set with a connection string. - Empty when never assigned.

Setter returns no value.

399.5 How It Works

- Assigning ActiveConnection updates the connection source used by Tables.
- Each assignment invalidates previously cached Tables metadata.

399.6 Remarks

- Property names are case-insensitive.
- Use Set when assigning an object value.

399.7 Example

```
<%
Option Explicit

Dim conn, catalog, current
Set conn = Server.CreateObject("ADODB.Connection")
conn.Open "Data Source=./temp/sample.db;Version=3;"

Set catalog = Server.CreateObject("ADOX.Catalog")
Set catalog.ActiveConnection = conn

Set current = catalog.ActiveConnection
If IsObject(current) Then
        Response.Write "ActiveConnection is an object"
Else
        Response.Write "ActiveConnection is not an object"
End If

Set current = Nothing
Set catalog = Nothing
conn.Close
Set conn = Nothing
%>
```

400 Get Catalog Tables

400.1 Overview

Use Tables to retrieve the catalog table collection for the current ActiveConnection.

400.2 Prerequisites

- Set ActiveConnection before reading Tables.

400.3 Syntax

```
Set tables = catalog.Tables
count = tables.Count
Set oneTable = tables.Item(0)
```

400.4 Parameters

- Getter only. This property accepts no arguments.

400.5 Return Value

Returns an object reference to an ADOX tables collection.

The collection exposes: - Count (Integer): number of discovered items. - Item(indexOrName) (Object or Empty): returns one table object by zero-based index or case-insensitive table name.

Each table object exposes: - Name (String) - Type (String)

400.6 How It Works

- On first read, the runtime resolves schema metadata from ActiveConnection and builds the collection.
- The collection is cached until ActiveConnection changes.

400.7 Remarks

- Tables is read-only.
- Use Set when assigning object returns.

400.8 Example

```
<%
Option Explicit

Dim conn, catalog, tables, firstTable
Set conn = Server.CreateObject("ADODB.Connection")
conn.Open "Data Source=./temp/sample.db;Version=3;"

Set catalog = Server.CreateObject("ADOX.Catalog")
Set catalog.ActiveConnection = conn
Set tables = catalog.Tables

Response.Write "Count=" & tables.Count & "<br>"
If tables.Count > 0 Then
    Set firstTable = tables.Item(0)
    Response.Write "First name=" & firstTable.Name & "<br>"
    Response.Write "First type=" & firstTable.Type
    Set firstTable = Nothing
End If

Set tables = Nothing
Set catalog = Nothing
conn.Close
```

```
Set conn = Nothing
%>
```

401 Use MSWC Components in G3Pix AxonASP

401.1 Overview

G3Pix AxonASP provides a compatibility surface for legacy MSWC-style components. Each component is exposed through native object dispatch with case-insensitive member names.

401.2 Supported ProgIDs

- MSWC.AdRotator
- MSWC.BrowserType
- MSWC.NextLink
- MSWC.ContentRotator
- MSWC.Counters
- MSWC.Tools
- MSWC.MyInfo
- MSWC.PageCounter
- MSWC.PermissionChecker

401.3 Prerequisites

- Instantiate components with Server.CreateObject using the exact ProgID.
- Ensure referenced files exist when using file-based components such as AdRotator, NextLink, and ContentRotator.
- Enable MSWC page counter configuration before using MSWC.PageCounter.

401.4 How It Works

- Each component has its own method and property surface.
- Unsupported members return Empty.
- Some components read request or server context to produce results.

401.5 Code Example

```
<%
Option Explicit

Dim browserType, browserName, canUseCookies
Set browserType = Server.CreateObject("MSWC.BrowserType")

browserName = browserType.Browser
canUseCookies = browserType.Cookies
```

```
Response.Write "Browser=" & browserName & " Cookies=" & CStr(canUseCookie
s)

Set browserType = Nothing
%>
```

402 MSWC Methods

402.1 Overview

This page summarizes methods exposed by MSWC compatibility components in G3Pix AxonASP.

402.2 Methods Reference

Component	Method	Returns	Description
MSWC.AdRotator	GetAdvertisement(scheduleFile)	String	Returns generated advertisement HTML. Returns an empty string when no argument is provided or no entries are available.
MSWC.NextLink	GetListCount(listFile)	Integer or Empty	Returns the number of entries in the link list. Returns Empty when listFile is missing.
MSWC.NextLink	GetListIndex(listFile)	Integer or Empty	Returns the current 1-based index that matches the executing page URL. Returns Empty when listFile is missing.
MSWC.NextLink	GetNextDescription(listFile)	String or Empty	Returns the description of the next entry. Returns Empty when listFile is missing.
MSWC.NextLink	GetNextURL(listFile)	String or Empty	Returns the URL of the next entry.

Component	Method	Returns	Description
			Returns Empty when listFile is missing.
MSWC.NextLink	GetPreviousDescription(listFile)	String or Empty	Returns the description of the previous entry. Returns Empty when listFile is missing.
MSWC.NextLink	GetPreviousURL(listFile)	String or Empty	Returns the URL of the previous entry. Returns Empty when listFile is missing.
MSWC.NextLink	GetNthDescription(listFile, index)	String or Empty	Returns the description of the specified 1-based entry. Returns Empty when required arguments are missing.
MSWC.NextLink	GetNthURL(listFile, index)	String or Empty	Returns the URL of the specified 1-based entry. Returns Empty when required arguments are missing.
MSWC.ContentRotator	ChooseContent(contentFile)	String or Empty	Returns one selected content block using weight rules. Returns Empty when contentFile is missing.
MSWC.ContentRotator	GetAllContent(contentFile)	String or Empty	Returns all content blocks joined by separator markup. Returns Empty

Component	Method	Returns	Description
			when contentFile is missing.
MSWC.Counters	Get(name)	Integer or Empty	Returns the current counter value or 0 when the counter does not exist. Returns Empty when name is missing.
MSWC.Counters	Increment(name)	Integer or Empty	Increments and returns the counter value. Returns Empty when name is missing.
MSWC.Counters	Remove(name)	Empty	Removes one counter key. Returns Empty.
MSWC.Counters	Set(name, value)	Empty	Sets one counter value when arguments are provided. Returns Empty.
MSWC.Tools	FileExists(path)	Boolean	Returns True when the mapped file exists, otherwise False.
MSWC.Tools	Owner(path)	String	Returns the resolved owner name for the mapped file path, or an empty string when unavailable.
MSWC.Tools	PluginExists(name)	Boolean	Returns False.
MSWC.Tools	ProcessForm(...)	Empty	Returns Empty.
MSWC.MyInfo	URL(index)	String or Empty	Returns URL-indexed value from MyInfo.xml, such as URL1 or URL2.

Component	Method	Returns	Description
			Returns Empty when not found or argument is missing.
MSWC.MyInfo	URLWords(index)	String or Empty	Returns URLWords-indexed value from MyInfo.xml, such as URLWords1. Returns Empty when not found or argument is missing.
MSWC.PageCounter	Hits([path])	Integer or Empty	Returns hit count for path or current script URL. Returns 0 when no path is resolved. Returns Empty when the component is disabled.
MSWC.PageCounter	PageHit()	Integer or Empty	Increments and returns hit count for the current script URL. Returns 0 when no path is resolved. Returns Empty when the component is disabled.
MSWC.PageCounter	Reset([path])	Empty	Removes page counter state for path or current script URL. Returns Empty.
MSWC.PermissionChecker	HasAccess(path)	Boolean or Empty	Returns True when the mapped path is readable by the process, otherwise False. Returns

Component	Method	Returns	Description
			Empty when path is missing.

402.3 Remarks

- Method names are case-insensitive.
- Methods that return Empty indicate missing required arguments or unsupported operation context for that call path.

403 MSWC Properties

403.1 Overview

This page summarizes property members exposed by MSWC compatibility components in G3Pix AxonASP.

403.2 Properties Reference

Component	Property	Access	Type	Description
MSWC.AdRotator	Border	Read/Write	Integer	Border value used in generated image HTML. Default is -1, which uses schedule-file border metadata.
MSWC.AdRotator	Clickable	Read/Write	Boolean	Controls whether generated advertisement HTML is wrapped in anchor tags.
MSWC.AdRotator	TargetFrame	Read/Write	String	Target frame name used when clickable advertisements are rendered.
MSWC.BrowserType	Browser	Read-only	String	Detected browser label,

Component	Property	Access	Type	Description
				such as IE, Edge, Chrome, Firefox, Safari, or Unknown.
MSWC.BrowserType	Version	Read-only	String	Browser version string. Current compatibility default is 0.0.
MSWC.BrowserType	MajorVer	Read-only	String	Browser major version string. Current compatibility default is 0.
MSWC.BrowserType	MinorVer	Read-only	String	Browser minor version string. Current compatibility default is 0.
MSWC.BrowserType	Frames	Read-only	Boolean	Indicates frame support based on detected user agent heuristics.
MSWC.BrowserType	Tables	Read-only	Boolean	Indicates HTML table support.
MSWC.BrowserType	Cookies	Read-only	Boolean	Indicates cookie support.
MSWC.BrowserType	BackgroundSounds	Read-only	Boolean	Indicates background sound support.
MSWC.BrowserType	VBScript	Read-only	Boolean	Indicates VBScript support.
MSWC.BrowserType	JavaScript	Read-only	Boolean	Indicates JavaScript

Component	Property	Access	Type	Description
				support.
MSWC.BrowserType	JavaApplets	Read-only	Boolean	Indicates Java applet support.
MSWC.BrowserType	ActiveXControls	Read-only	Boolean	Indicates ActiveX control support.
MSWC.BrowserType	CDF	Read-only	Boolean	Indicates CDF support.
MSWC.MyInfo	AnyCustomProperty	Read-only	String or Empty	Returns one property value loaded from MyInfo.xml when the property key exists, otherwise Empty.

403.3 Remarks

- Property names are case-insensitive.
- Unsupported property names resolve through component dispatch and may return Empty.

404 AdRotator.GetAdvertisement Method

404.1 Overview

Calls the GetAdvertisement member on the AdRotator compatibility object.

404.2 Syntax

```
Dim obj
Set obj = Server.CreateObject("MSWC.AdRotator")
html = obj.GetAdvertisement(scheduleFile)
```

404.3 Parameters and Arguments

- Parameters are validated by runtime dispatch for MSWC.AdRotator.
- Invalid argument count or incompatible values can raise runtime errors.

404.4 Return Values

Returns a Variant-compatible value according to operation semantics.

404.5 Remarks

- Reads an ad schedule file and returns the selected ad HTML fragment.
- Use Set when assigning object return values.
- Member names are case-insensitive.

404.6 Code Example

```
<%
Dim obj, result
Set obj = Server.CreateObject("MSWC.AdRotator")
result = Empty
On Error Resume Next
result = obj.GetAdvertisement()
On Error GoTo 0
Response.Write CStr(result)
Set obj = Nothing
%>
```

405 NextLink.GetListCount Method

405.1 Overview

Calls the GetListCount member on the NextLink compatibility object.

405.2 Syntax

```
Dim obj
Set obj = Server.CreateObject("MSWC.NextLink")
n = obj.GetListCount(listFile)
```

405.3 Parameters and Arguments

- Parameters are validated by runtime dispatch for MSWC.NextLink.
- Invalid argument count or incompatible values can raise runtime errors.

405.4 Return Values

Returns a Variant-compatible value according to operation semantics.

405.5 Remarks

- Returns number of entries in the link list file.
- Use Set when assigning object return values.
- Member names are case-insensitive.

405.6 Code Example

```
<%
Dim obj, result
Set obj = Server.CreateObject("MSWC.NextLink")
result = Empty
On Error Resume Next
result = obj.GetListCount()
On Error GoTo 0
Response.Write CStr(result)
Set obj = Nothing
%>
```

406 NextLink.GetListIndex Method

406.1 Overview

Calls the GetListIndex member on the NextLink compatibility object.

406.2 Syntax

```
Dim obj
Set obj = Server.CreateObject("MSWC.NextLink")
n = obj.GetListIndex(listFile)
```

406.3 Parameters and Arguments

- Parameters are validated by runtime dispatch for MSWC.NextLink.
- Invalid argument count or incompatible values can raise runtime errors.

406.4 Return Values

Returns a Variant-compatible value according to operation semantics.

406.5 Remarks

- Returns current item index based on current request URL.
- Use Set when assigning object return values.
- Member names are case-insensitive.

406.6 Code Example

```
<%
Dim obj, result
Set obj = Server.CreateObject("MSWC.NextLink")
result = Empty
On Error Resume Next
result = obj.GetListIndex()
On Error GoTo 0
Response.Write CStr(result)
```

```
Set obj = Nothing
%>
```

407 NextLink.GetNextDescription Method

407.1 Overview

Calls the GetNextDescription member on the NextLink compatibility object.

407.2 Syntax

```
Dim obj
Set obj = Server.CreateObject("MSWC.NextLink")
text = obj.GetNextDescription(listFile)
```

407.3 Parameters and Arguments

- Parameters are validated by runtime dispatch for MSWC.NextLink.
- Invalid argument count or incompatible values can raise runtime errors.

407.4 Return Values

Returns a Variant-compatible value according to operation semantics.

407.5 Remarks

- Returns description of next entry.
- Use Set when assigning object return values.
- Member names are case-insensitive.

407.6 Code Example

```
<%
Dim obj, result
Set obj = Server.CreateObject("MSWC.NextLink")
result = Empty
On Error Resume Next
result = obj.GetNextDescription()
On Error GoTo 0
Response.Write CStr(result)
Set obj = Nothing
%>
```

408 NextLink.GetNextURL Method

408.1 Overview

Calls the GetNextURL member on the NextLink compatibility object.

408.2 Syntax

```
Dim obj
Set obj = Server.CreateObject("MSWC.NextLink")
url = obj.GetNextURL(listFile)
```

408.3 Parameters and Arguments

- Parameters are validated by runtime dispatch for MSWC.NextLink.
- Invalid argument count or incompatible values can raise runtime errors.

408.4 Return Values

Returns a Variant-compatible value according to operation semantics.

408.5 Remarks

- Returns URL of next entry.
- Use Set when assigning object return values.
- Member names are case-insensitive.

408.6 Code Example

```
<%
Dim obj, result
Set obj = Server.CreateObject("MSWC.NextLink")
result = Empty
On Error Resume Next
result = obj.GetNextURL()
On Error GoTo 0
Response.Write CStr(result)
Set obj = Nothing
%>
```

409 NextLink.GetPreviousDescription Method

409.1 Overview

Calls the GetPreviousDescription member on the NextLink compatibility object.

409.2 Syntax

```
Dim obj
Set obj = Server.CreateObject("MSWC.NextLink")
text = obj.GetPreviousDescription(listFile)
```

409.3 Parameters and Arguments

- Parameters are validated by runtime dispatch for MSWC.NextLink.
- Invalid argument count or incompatible values can raise runtime errors.

409.4 Return Values

Returns a Variant-compatible value according to operation semantics.

409.5 Remarks

- Returns description of previous entry.
- Use Set when assigning object return values.
- Member names are case-insensitive.

409.6 Code Example

```
<%
Dim obj, result
Set obj = Server.CreateObject("MSWC.NextLink")
result = Empty
On Error Resume Next
result = obj.GetPreviousDescription()
On Error GoTo 0
Response.Write CStr(result)
Set obj = Nothing
%>
```

410 NextLink.GetPreviousURL Method

410.1 Overview

Calls the GetPreviousURL member on the NextLink compatibility object.

410.2 Syntax

```
Dim obj
Set obj = Server.CreateObject("MSWC.NextLink")
url = obj.GetPreviousURL(listFile)
```

410.3 Parameters and Arguments

- Parameters are validated by runtime dispatch for MSWC.NextLink.
- Invalid argument count or incompatible values can raise runtime errors.

410.4 Return Values

Returns a Variant-compatible value according to operation semantics.

410.5 Remarks

- Returns URL of previous entry.
- Use Set when assigning object return values.
- Member names are case-insensitive.

410.6 Code Example

```
<%
Dim obj, result
Set obj = Server.CreateObject("MSWC.NextLink")
result = Empty
On Error Resume Next
result = obj.GetPreviousURL()
On Error GoTo 0
Response.Write CStr(result)
Set obj = Nothing
%>
```

411 NextLink.GetNthDescription Method

411.1 Overview

Calls the GetNthDescription member on the NextLink compatibility object.

411.2 Syntax

```
Dim obj
Set obj = Server.CreateObject("MSWC.NextLink")
text = obj.GetNthDescription(listFile, index)
```

411.3 Parameters and Arguments

- Parameters are validated by runtime dispatch for MSWC.NextLink.
- Invalid argument count or incompatible values can raise runtime errors.

411.4 Return Values

Returns a Variant-compatible value according to operation semantics.

411.5 Remarks

- Returns description for 1-based list index.
- Use Set when assigning object return values.
- Member names are case-insensitive.

411.6 Code Example

```
<%
Dim obj, result
Set obj = Server.CreateObject("MSWC.NextLink")
result = Empty
On Error Resume Next
result = obj.GetNthDescription()
On Error GoTo 0
Response.Write CStr(result)
```

```
Set obj = Nothing
%>
```

412 NextLink.GetNthURL Method

412.1 Overview

Calls the GetNthURL member on the NextLink compatibility object.

412.2 Syntax

```
Dim obj
Set obj = Server.CreateObject("MSWC.NextLink")
url = obj.GetNthURL(listFile, index)
```

412.3 Parameters and Arguments

- Parameters are validated by runtime dispatch for MSWC.NextLink.
- Invalid argument count or incompatible values can raise runtime errors.

412.4 Return Values

Returns a Variant-compatible value according to operation semantics.

412.5 Remarks

- Returns URL for 1-based list index.
- Use Set when assigning object return values.
- Member names are case-insensitive.

412.6 Code Example

```
<%
Dim obj, result
Set obj = Server.CreateObject("MSWC.NextLink")
result = Empty
On Error Resume Next
result = obj.GetNthURL()
On Error GoTo 0
Response.Write CStr(result)
Set obj = Nothing
%>
```

413 ContentRotator.ChooseContent Method

413.1 Overview

Calls the ChooseContent member on the ContentRotator compatibility object.

413.2 Syntax

```
Dim obj
Set obj = Server.CreateObject("MSWC.ContentRotator")
html = obj.ChooseContent(contentFile)
```

413.3 Parameters and Arguments

- Parameters are validated by runtime dispatch for MSWC.ContentRotator.
- Invalid argument count or incompatible values can raise runtime errors.

413.4 Return Values

Returns a Variant-compatible value according to operation semantics.

413.5 Remarks

- Returns one weighted content block from rotator file.
- Use Set when assigning object return values.
- Member names are case-insensitive.

413.6 Code Example

```
<%
Dim obj, result
Set obj = Server.CreateObject("MSWC.ContentRotator")
result = Empty
On Error Resume Next
result = obj.ChooseContent()
On Error GoTo 0
Response.Write CStr(result)
Set obj = Nothing
%>
```

414 ContentRotator.GetAllContent Method

414.1 Overview

Calls the GetAllContent member on the ContentRotator compatibility object.

414.2 Syntax

```
Dim obj
Set obj = Server.CreateObject("MSWC.ContentRotator")
html = obj.GetAllContent(contentFile)
```

414.3 Parameters and Arguments

- Parameters are validated by runtime dispatch for MSWC.ContentRotator.
- Invalid argument count or incompatible values can raise runtime errors.

414.4 Return Values

Returns a Variant-compatible value according to operation semantics.

414.5 Remarks

- Returns all content blocks joined with separators.
- Use Set when assigning object return values.
- Member names are case-insensitive.

414.6 Code Example

```
<%
Dim obj, result
Set obj = Server.CreateObject("MSWC.ContentRotator")
result = Empty
On Error Resume Next
result = obj.GetAllContent()
On Error GoTo 0
Response.Write CStr(result)
Set obj = Nothing
%>
```

415 Counters.Get Method

415.1 Overview

Calls the Get member on the Counters compatibility object.

415.2 Syntax

```
Dim obj
Set obj = Server.CreateObject("MSWC.Counters")
n = obj.Get(counterName)
```

415.3 Parameters and Arguments

- Parameters are validated by runtime dispatch for MSWC.Counters.
- Invalid argument count or incompatible values can raise runtime errors.

415.4 Return Values

Returns a Variant-compatible value according to operation semantics.

415.5 Remarks

- Gets integer value of a named counter.
- Use Set when assigning object return values.
- Member names are case-insensitive.

415.6 Code Example

```
<%
Dim obj, result
Set obj = Server.CreateObject("MSWC.Counters")
result = Empty
On Error Resume Next
result = obj.Get()
On Error GoTo 0
Response.Write CStr(result)
Set obj = Nothing
%>
```

416 Counters.Increment Method

416.1 Overview

Calls the Increment member on the Counters compatibility object.

416.2 Syntax

```
Dim obj
Set obj = Server.CreateObject("MSWC.Counters")
n = obj.Increment(counterName)
```

416.3 Parameters and Arguments

- Parameters are validated by runtime dispatch for MSWC.Counters.
- Invalid argument count or incompatible values can raise runtime errors.

416.4 Return Values

Returns a Variant-compatible value according to operation semantics.

416.5 Remarks

- Increments counter and returns new value.
- Use Set when assigning object return values.
- Member names are case-insensitive.

416.6 Code Example

```
<%
Dim obj, result
Set obj = Server.CreateObject("MSWC.Counters")
result = Empty
On Error Resume Next
result = obj.Increment()
On Error GoTo 0
Response.Write CStr(result)
```

```
Set obj = Nothing
%>
```

417 Counters.Remove Method

417.1 Overview

Calls the Remove member on the Counters compatibility object.

417.2 Syntax

```
Dim obj
Set obj = Server.CreateObject("MSWC.Counters")
obj.Remove counterName
```

417.3 Parameters and Arguments

- Parameters are validated by runtime dispatch for MSWC.Counters.
- Invalid argument count or incompatible values can raise runtime errors.

417.4 Return Values

Returns a Variant-compatible value according to operation semantics.

417.5 Remarks

- Deletes named counter entry.
- Use Set when assigning object return values.
- Member names are case-insensitive.

417.6 Code Example

```
<%
Dim obj, result
Set obj = Server.CreateObject("MSWC.Counters")
result = Empty
On Error Resume Next
result = obj.Remove()
On Error GoTo 0
Response.Write CStr(result)
Set obj = Nothing
%>
```

418 Counters.Set Method

418.1 Overview

Calls the Set member on the Counters compatibility object.

418.2 Syntax

```
Dim obj
Set obj = Server.CreateObject("MSWC.Counters")
obj.Set counterName, value
```

418.3 Parameters and Arguments

- Parameters are validated by runtime dispatch for MSWC.Counters.
- Invalid argument count or incompatible values can raise runtime errors.

418.4 Return Values

Returns a Variant-compatible value according to operation semantics.

418.5 Remarks

- Sets named counter to integer value.
- Use Set when assigning object return values.
- Member names are case-insensitive.

418.6 Code Example

```
<%
Dim obj, result
Set obj = Server.CreateObject("MSWC.Counters")
result = Empty
On Error Resume Next
result = obj.Set()
On Error GoTo 0
Response.Write CStr(result)
Set obj = Nothing
%>
```

419 Tools.FileExists Method

419.1 Overview

Calls the FileExists member on the Tools compatibility object.

419.2 Syntax

```
Dim obj
Set obj = Server.CreateObject("MSWC.Tools")
ok = obj.FileExists(virtualPath)
```

419.3 Parameters and Arguments

- Parameters are validated by runtime dispatch for MSWC.Tools.
- Invalid argument count or incompatible values can raise runtime errors.

419.4 Return Values

Returns a Variant-compatible value according to operation semantics.

419.5 Remarks

- Returns True when mapped file exists.
- Use Set when assigning object return values.
- Member names are case-insensitive.

419.6 Code Example

```
<%
Dim obj, result
Set obj = Server.CreateObject("MSWC.Tools")
result = Empty
On Error Resume Next
result = obj.FileExists()
On Error GoTo 0
Response.Write CStr(result)
Set obj = Nothing
%>
```

420 Tools.Owner Method

420.1 Overview

Calls the Owner member on the Tools compatibility object.

420.2 Syntax

```
Dim obj
Set obj = Server.CreateObject("MSWC.Tools")
name = obj.Owner(virtualPath)
```

420.3 Parameters and Arguments

- Parameters are validated by runtime dispatch for MSWC.Tools.
- Invalid argument count or incompatible values can raise runtime errors.

420.4 Return Values

Returns a Variant-compatible value according to operation semantics.

420.5 Remarks

- Returns best-effort file owner name.
- Use Set when assigning object return values.
- Member names are case-insensitive.

420.6 Code Example

```
<%
Dim obj, result
Set obj = Server.CreateObject("MSWC.Tools")
result = Empty
On Error Resume Next
result = obj.Owner()
On Error GoTo 0
Response.Write CStr(result)
Set obj = Nothing
%>
```

421 Tools.PluginExists Method

421.1 Overview

Calls the PluginExists member on the Tools compatibility object.

421.2 Syntax

```
Dim obj
Set obj = Server.CreateObject("MSWC.Tools")
ok = obj.PluginExists(pluginName)
```

421.3 Parameters and Arguments

- Parameters are validated by runtime dispatch for MSWC.Tools.
- Invalid argument count or incompatible values can raise runtime errors.

421.4 Return Values

Returns a Variant-compatible value according to operation semantics.

421.5 Remarks

- Compatibility stub currently returns False.
- Use Set when assigning object return values.
- Member names are case-insensitive.

421.6 Code Example

```
<%
Dim obj, result
Set obj = Server.CreateObject("MSWC.Tools")
result = Empty
On Error Resume Next
result = obj.PluginExists()
On Error GoTo 0
Response.Write CStr(result)
```

```
Set obj = Nothing
%>
```

422 Tools.ProcessForm Method

422.1 Overview

Calls the ProcessForm member on the Tools compatibility object.

422.2 Syntax

```
Dim obj
Set obj = Server.CreateObject("MSWC.Tools")
obj.ProcessForm
```

422.3 Parameters and Arguments

- Parameters are validated by runtime dispatch for MSWC.Tools.
- Invalid argument count or incompatible values can raise runtime errors.

422.4 Return Values

Returns a Variant-compatible value according to operation semantics.

422.5 Remarks

- Compatibility stub; currently no-op.
- Use Set when assigning object return values.
- Member names are case-insensitive.

422.6 Code Example

```
<%
Dim obj, result
Set obj = Server.CreateObject("MSWC.Tools")
result = Empty
On Error Resume Next
result = obj.ProcessForm()
On Error GoTo 0
Response.Write CStr(result)
Set obj = Nothing
%>
```

423 MyInfo.URL Method

423.1 Overview

Calls the URL member on the MyInfo compatibility object.

423.2 Syntax

```
Dim obj
Set obj = Server.CreateObject("MSWC.MyInfo")
value = obj.URL(index)
```

423.3 Parameters and Arguments

- Parameters are validated by runtime dispatch for MSWC.MyInfo.
- Invalid argument count or incompatible values can raise runtime errors.

423.4 Return Values

Returns a Variant-compatible value according to operation semantics.

423.5 Remarks

- Returns URL field value by suffix index from MyInfo.xml.
- Use Set when assigning object return values.
- Member names are case-insensitive.

423.6 Code Example

```
<%
Dim obj, result
Set obj = Server.CreateObject("MSWC.MyInfo")
result = Empty
On Error Resume Next
result = obj.URL()
On Error GoTo 0
Response.Write CStr(result)
Set obj = Nothing
%>
```

424 MyInfo.URLWords Method

424.1 Overview

Calls the URLWords member on the MyInfo compatibility object.

424.2 Syntax

```
Dim obj
Set obj = Server.CreateObject("MSWC.MyInfo")
value = obj.URLWords(index)
```

424.3 Parameters and Arguments

- Parameters are validated by runtime dispatch for MSWC.MyInfo.
- Invalid argument count or incompatible values can raise runtime errors.

424.4 Return Values

Returns a Variant-compatible value according to operation semantics.

424.5 Remarks

- Returns URLWords field value by suffix index from MyInfo.xml.
- Use Set when assigning object return values.
- Member names are case-insensitive.

424.6 Code Example

```
<%
Dim obj, result
Set obj = Server.CreateObject("MSWC.MyInfo")
result = Empty
On Error Resume Next
result = obj.URLWords()
On Error GoTo 0
Response.Write CStr(result)
Set obj = Nothing
%>
```

425 PageCounter.Hits Method

425.1 Overview

Calls the Hits member on the PageCounter compatibility object.

425.2 Syntax

```
Dim obj
Set obj = Server.CreateObject("MSWC.PageCounter")
n = obj.Hits([path])
```

425.3 Parameters and Arguments

- Parameters are validated by runtime dispatch for MSWC.PageCounter.
- Invalid argument count or incompatible values can raise runtime errors.

425.4 Return Values

Returns a Variant-compatible value according to operation semantics.

425.5 Remarks

- Reads hit count for current page or provided path.
- Use Set when assigning object return values.
- Member names are case-insensitive.

425.6 Code Example

```
<%
Dim obj, result
Set obj = Server.CreateObject("MSWC.PageCounter")
result = Empty
On Error Resume Next
result = obj.Hits()
On Error GoTo 0
Response.Write CStr(result)
Set obj = Nothing
%>
```

426 PageCounter.PageHit Method

426.1 Overview

Calls the PageHit member on the PageCounter compatibility object.

426.2 Syntax

```
Dim obj
Set obj = Server.CreateObject("MSWC.PageCounter")
n = obj.PageHit()
```

426.3 Parameters and Arguments

- Parameters are validated by runtime dispatch for MSWC.PageCounter.
- Invalid argument count or incompatible values can raise runtime errors.

426.4 Return Values

Returns a Variant-compatible value according to operation semantics.

426.5 Remarks

- Increments and returns current page hit count.
- Use Set when assigning object return values.
- Member names are case-insensitive.

426.6 Code Example

```
<%
Dim obj, result
Set obj = Server.CreateObject("MSWC.PageCounter")
result = Empty
On Error Resume Next
result = obj.PageHit()
On Error GoTo 0
Response.Write CStr(result)
```

```
Set obj = Nothing
%>
```

427 PageCounter.Reset Method

427.1 Overview

Calls the Reset member on the PageCounter compatibility object.

427.2 Syntax

```
Dim obj
Set obj = Server.CreateObject("MSWC.PageCounter")
obj.Reset [path]
```

427.3 Parameters and Arguments

- Parameters are validated by runtime dispatch for MSWC.PageCounter.
- Invalid argument count or incompatible values can raise runtime errors.

427.4 Return Values

Returns a Variant-compatible value according to operation semantics.

427.5 Remarks

- Resets current page or provided path counter.
- Use Set when assigning object return values.
- Member names are case-insensitive.

427.6 Code Example

```
<%
Dim obj, result
Set obj = Server.CreateObject("MSWC.PageCounter")
result = Empty
On Error Resume Next
result = obj.Reset()
On Error GoTo 0
Response.Write CStr(result)
Set obj = Nothing
%>
```

428 PermissionChecker.HasAccess Method

428.1 Overview

Calls the HasAccess member on the PermissionChecker compatibility object.

428.2 Syntax

```
Dim obj
Set obj = Server.CreateObject("MSWC.PermissionChecker")
ok = obj.HasAccess(virtualPath)
```

428.3 Parameters and Arguments

- Parameters are validated by runtime dispatch for MSWC.PermissionChecker.
- Invalid argument count or incompatible values can raise runtime errors.

428.4 Return Values

Returns a Variant-compatible value according to operation semantics.

428.5 Remarks

- Returns True when process can open mapped path for reading.
- Use Set when assigning object return values.
- Member names are case-insensitive.

428.6 Code Example

```
<%
Dim obj, result
Set obj = Server.CreateObject("MSWC.PermissionChecker")
result = Empty
On Error Resume Next
result = obj.HasAccess()
On Error GoTo 0
Response.Write CStr(result)
Set obj = Nothing
%>
```

429 AdRotator.Border Property

429.1 Overview

Reads or writes the Border member on the AdRotator compatibility object.

429.2 Syntax

```
Dim obj, value
Set obj = Server.CreateObject("MSWC.AdRotator")
value = obj.Border
```

429.3 Access

Read/Write

429.4 Return Values

Returns a Variant-compatible value.

429.5 Remarks

- Overrides border value used by generated IMG tag.
- Property names are case-insensitive.

429.6 Code Example

```
<%
Dim obj
Set obj = Server.CreateObject("MSWC.AdRotator")
Response.Write CStr(obj.Border)
Set obj = Nothing
%>
```

430 AdRotator.Clickable Property

430.1 Overview

Reads or writes the Clickable member on the AdRotator compatibility object.

430.2 Syntax

```
Dim obj, value
Set obj = Server.CreateObject("MSWC.AdRotator")
value = obj.Clickable
```

430.3 Access

Read/Write

430.4 Return Values

Returns a Variant-compatible value.

430.5 Remarks

- Controls if generated output wraps IMG in anchor tag.
- Property names are case-insensitive.

430.6 Code Example

```
<%
Dim obj
Set obj = Server.CreateObject("MSWC.AdRotator")
Response.Write CStr(obj.Clickable)
Set obj = Nothing
%>
```

431 AdRotator.TargetFrame Property

431.1 Overview

Reads or writes the TargetFrame member on the AdRotator compatibility object.

431.2 Syntax

```
Dim obj, value
Set obj = Server.CreateObject("MSWC.AdRotator")
value = obj.TargetFrame
```

431.3 Access

Read/Write

431.4 Return Values

Returns a Variant-compatible value.

431.5 Remarks

- Sets optional target frame on generated anchor element.
- Property names are case-insensitive.

431.6 Code Example

```
<%
Dim obj
Set obj = Server.CreateObject("MSWC.AdRotator")
Response.Write CStr(obj.TargetFrame)
Set obj = Nothing
%>
```

432 BrowserType.Browser Property

432.1 Overview

Reads or writes the Browser member on the BrowserType compatibility object.

432.2 Syntax

```
Dim obj, value
Set obj = Server.CreateObject("MSWC.BrowserType")
value = obj.Browser
```

432.3 Access

Read Only

432.4 Return Values

Returns a Variant-compatible value.

432.5 Remarks

- Detected browser family name.
- Property names are case-insensitive.

432.6 Code Example

```
<%
Dim obj
Set obj = Server.CreateObject("MSWC.BrowserType")
Response.Write CStr(obj.Browser)
Set obj = Nothing
%>
```

433 BrowserType.Version Property

433.1 Overview

Reads or writes the Version member on the BrowserType compatibility object.

433.2 Syntax

```
Dim obj, value
Set obj = Server.CreateObject("MSWC.BrowserType")
value = obj.Version
```

433.3 Access

Read Only

433.4 Return Values

Returns a Variant-compatible value.

433.5 Remarks

- Detected browser version string.
- Property names are case-insensitive.

433.6 Code Example

```
<%
Dim obj
Set obj = Server.CreateObject("MSWC.BrowserType")
Response.Write CStr(obj.Version)
Set obj = Nothing
%>
```

434 BrowserType.MajorVer Property

434.1 Overview

Reads or writes the MajorVer member on the BrowserType compatibility object.

434.2 Syntax

```
Dim obj, value
Set obj = Server.CreateObject("MSWC.BrowserType")
value = obj.MajorVer
```

434.3 Access

Read Only

434.4 Return Values

Returns a Variant-compatible value.

434.5 Remarks

- Detected major version string.
- Property names are case-insensitive.

434.6 Code Example

```
<%
Dim obj
Set obj = Server.CreateObject("MSWC.BrowserType")
Response.Write CStr(obj.MajorVer)
Set obj = Nothing
%>
```

435 BrowserType.MinorVer Property

435.1 Overview

Reads or writes the MinorVer member on the BrowserType compatibility object.

435.2 Syntax

```
Dim obj, value
Set obj = Server.CreateObject("MSWC.BrowserType")
value = obj.MinorVer
```

435.3 Access

Read Only

435.4 Return Values

Returns a Variant-compatible value.

435.5 Remarks

- Detected minor version string.
- Property names are case-insensitive.

435.6 Code Example

```
<%
Dim obj
Set obj = Server.CreateObject("MSWC.BrowserType")
Response.Write CStr(obj.MinorVer)
Set obj = Nothing
%>
```

436 BrowserType.Frames Property

436.1 Overview

Reads or writes the Frames member on the BrowserType compatibility object.

436.2 Syntax

```
Dim obj, value
Set obj = Server.CreateObject("MSWC.BrowserType")
value = obj.Frames
```

436.3 Access

Read Only

436.4 Return Values

Returns a Variant-compatible value.

436.5 Remarks

- Indicates frames capability.
- Property names are case-insensitive.

436.6 Code Example

```
<%
Dim obj
Set obj = Server.CreateObject("MSWC.BrowserType")
Response.Write CStr(obj.Frames)
Set obj = Nothing
%>
```

437 BrowserType.Tables Property

437.1 Overview

Reads or writes the Tables member on the BrowserType compatibility object.

437.2 Syntax

```
Dim obj, value
Set obj = Server.CreateObject("MSWC.BrowserType")
value = obj.Tables
```

437.3 Access

Read Only

437.4 Return Values

Returns a Variant-compatible value.

437.5 Remarks

- Indicates tables capability.
- Property names are case-insensitive.

437.6 Code Example

```
<%
Dim obj
Set obj = Server.CreateObject("MSWC.BrowserType")
Response.Write CStr(obj.Tables)
Set obj = Nothing
%>
```

438 BrowserType.Cookies Property

438.1 Overview

Reads or writes the Cookies member on the BrowserType compatibility object.

438.2 Syntax

```
Dim obj, value
Set obj = Server.CreateObject("MSWC.BrowserType")
value = obj.Cookies
```

438.3 Access

Read Only

438.4 Return Values

Returns a Variant-compatible value.

438.5 Remarks

- Indicates cookies capability.
- Property names are case-insensitive.

438.6 Code Example

```
<%
Dim obj
Set obj = Server.CreateObject("MSWC.BrowserType")
Response.Write CStr(obj.Cookies)
Set obj = Nothing
%>
```

439 BrowserType.BackgroundSounds Property

439.1 Overview

Reads or writes the BackgroundSounds member on the BrowserType compatibility object.

439.2 Syntax

```
Dim obj, value
Set obj = Server.CreateObject("MSWC.BrowserType")
value = obj.BackgroundSounds
```

439.3 Access

Read Only

439.4 Return Values

Returns a Variant-compatible value.

439.5 Remarks

- Indicates background sound capability.
- Property names are case-insensitive.

439.6 Code Example

```
<%
Dim obj
Set obj = Server.CreateObject("MSWC.BrowserType")
Response.Write CStr(obj.BackgroundSounds)
```

```
Set obj = Nothing
%>
```

440 BrowserType.VBScript Property

440.1 Overview

Reads or writes the VBScript member on the BrowserType compatibility object.

440.2 Syntax

```
Dim obj, value
Set obj = Server.CreateObject("MSWC.BrowserType")
value = obj.VBScript
```

440.3 Access

Read Only

440.4 Return Values

Returns a Variant-compatible value.

440.5 Remarks

- Indicates VBScript support capability.
- Property names are case-insensitive.

440.6 Code Example

```
<%
Dim obj
Set obj = Server.CreateObject("MSWC.BrowserType")
Response.Write CStr(obj.VBScript)
Set obj = Nothing
%>
```

441 BrowserType.JavaScript Property

441.1 Overview

Reads or writes the JavaScript member on the BrowserType compatibility object.

441.2 Syntax

```
Dim obj, value
Set obj = Server.CreateObject("MSWC.BrowserType")
value = obj.JavaScript
```

441.3 Access

Read Only

441.4 Return Values

Returns a Variant-compatible value.

441.5 Remarks

- Indicates JavaScript capability.
- Property names are case-insensitive.

441.6 Code Example

```
<%
Dim obj
Set obj = Server.CreateObject("MSWC.BrowserType")
Response.Write CStr(obj.JavaScript)
Set obj = Nothing
%>
```

442 BrowserType.JavaApplets Property

442.1 Overview

Reads or writes the JavaApplets member on the BrowserType compatibility object.

442.2 Syntax

```
Dim obj, value
Set obj = Server.CreateObject("MSWC.BrowserType")
value = obj.JavaApplets
```

442.3 Access

Read Only

442.4 Return Values

Returns a Variant-compatible value.

442.5 Remarks

- Indicates Java applet capability.
- Property names are case-insensitive.

442.6 Code Example

```
<%
Dim obj
```

```
Set obj = Server.CreateObject("MSWC.BrowserType")
Response.Write CStr(obj.JavaApplets)
Set obj = Nothing
%>
```

443 BrowserType.ActiveXControls Property

443.1 Overview

Reads or writes the ActiveXControls member on the BrowserType compatibility object.

443.2 Syntax

```
Dim obj, value
Set obj = Server.CreateObject("MSWC.BrowserType")
value = obj.ActiveXControls
```

443.3 Access

Read Only

443.4 Return Values

Returns a Variant-compatible value.

443.5 Remarks

- Indicates ActiveX capability.
- Property names are case-insensitive.

443.6 Code Example

```
<%
Dim obj
Set obj = Server.CreateObject("MSWC.BrowserType")
Response.Write CStr(obj.ActiveXControls)
Set obj = Nothing
%>
```

444 BrowserType.CDF Property

444.1 Overview

Reads or writes the CDF member on the BrowserType compatibility object.

444.2 Syntax

```
Dim obj, value
Set obj = Server.CreateObject("MSWC.BrowserType")
value = obj.CDF
```

444.3 Access

Read Only

444.4 Return Values

Returns a Variant-compatible value.

444.5 Remarks

- Indicates CDF capability flag.
- Property names are case-insensitive.

444.6 Code Example

```
<%
Dim obj
Set obj = Server.CreateObject("MSWC.BrowserType")
Response.Write CStr(obj.CDF)
Set obj = Nothing
%>
```

445 MyInfo.AnyCustomProperty Property

445.1 Overview

Reads or writes the AnyCustomProperty member on the MyInfo compatibility object.

445.2 Syntax

```
Dim obj, value
Set obj = Server.CreateObject("MSWC.MyInfo")
value = obj.AnyCustomProperty
```

445.3 Access

Read Only

445.4 Return Values

Returns a Variant-compatible value.

445.5 Remarks

- Dynamic properties are loaded from MyInfo.xml element names.
- Property names are case-insensitive.

445.6 Code Example

```
<%
Dim obj
```

```
Set obj = Server.CreateObject("MSWC.MyInfo")
Response.Write CStr(obj.AnyCustomProperty)
Set obj = Nothing
%>
```

446 ADODB Library

The ADODB library provides database connectivity and data manipulation through Connection, Recordset, Command, and supporting objects. Use these objects to execute SQL queries, manage transactions, and perform CRUD operations on any ODBC- or OLE DB-compatible database.

446.1 Supported Objects

- **Connection** — Manages database connections, executes SQL commands, begins transactions, and retrieves database schema information.
- **Recordset** — Represents a result set of rows and columns, with support for navigation, editing, deletion, and persistence.
- **Command** — Encapsulates SQL statements and stored procedure calls with parameter support.
- **Error** — Represents a single database error within the Errors collection.
- **Field** — Represents a single column within a Fields collection, including its metadata and value.
- **Parameter** — Represents a single SQL parameter within the Parameters collection.

446.2 Supported ProgIDs

ProgID	Object
`ADODB.Connection`	Database connection
`ADODB.Recordset`	Query result set
`ADODB.Command`	SQL statement or stored procedure
`ADODBOLE.Connection`	OLE DB connection (compatibility alias)

446.3 Prerequisites

None. No external installation is required beyond a valid ODBC driver or provider for your target database.

446.4 How It Works

Create a Connection object, configure its ConnectionString with the provider and data source, then call Open. Use the Connection to execute queries via the Execute method or bind it to Command and Recordset objects. Navigate Recordset rows using

MoveNext, MoveFirst, MoveLast, MovePrevious, or the Seek method. Modify rows with AddNew, Update, and Delete. Wrap multiple operations in explicit BeginTrans/CommitTrans blocks for transaction safety.

446.5 Code Example

```
<%
Option Explicit
Dim conn, rs

Set conn = Server.CreateObject("ADODB.Connection")
conn.ConnectionString = "Driver={SQLite3};Data Source=" & Server.MapPath
("./db.sqlite")
conn.Open

Set rs = conn.Execute("SELECT * FROM users")
Do While Not rs.EOF
    Response.Write rs.Fields("name").Value & "<br>"
    rs.MoveNext
Loop
rs.Close
conn.Close

Set rs = Nothing
Set conn = Nothing
%>
```

447 ADODB Methods Reference

447.1 Connection Methods

Method	Returns	Description
Open	Empty	Establishes a connection to a database using the configured ConnectionString.
Close	Empty	Closes the database connection.
Execute	Recordset, Integer, or Empty	Executes a SQL statement. Returns a Recordset for queries (SELECT, SHOW, PRAGMA), an Integer row count for modifications (INSERT, UPDATE, DELETE), or Empty on failure.

Method	Returns	Description
BeginTrans	Integer	Begins a transaction. Returns 1 on success, 0 on failure.
CommitTrans	Empty	Commits the current transaction.
RollbackTrans	Empty	Rolls back the current transaction.
OpenSchema	Recordset	Returns a Recordset containing database schema information (tables, columns, indexes, views, procedures, or foreign keys).
Cancel	Empty	Cancels the current operation (no-op in current runtime).

447.2 Recordset Methods

Method	Returns	Description
Open	Empty	Opens a result set from a SQL statement or Connection. Optionally sets CursorType, LockType, and other parameters.
Close	Empty	Closes the result set and releases database resources.
MoveNext	Empty	Moves to the next row. EOF becomes true if no next row exists.
MovePrevious	Empty	Moves to the previous row. BOF becomes true if no previous row exists.
MoveFirst	Empty	Moves to the first row.
MoveLast	Empty	Moves to the last row.
Move	Empty	Moves forward or backward by the specified

Method	Returns	Description
		row offset.
Cancel	Empty	Cancels the current operation (no-op in current runtime).
CancelBatch	Empty	Cancels pending batch updates (no-op in current runtime).
CancelUpdate	Empty	Cancels pending row updates not yet committed.
Find	Empty	Locates and moves to the first row matching the criteria (format: `field = value`).
NextRecordset	Recordset or Nothing	Returns the next result set if the command produced multiple result sets, or Nothing if no additional result set exists.
Clone	Recordset	Creates an independent copy of the current Recordset with the same structure.
CompareBookmarks	Integer	Compares two bookmarks. Returns -1 (first is less), 0 (equal), or 1 (first is greater).
AddNew	Empty	Adds a new empty row to the Recordset in memory. Follow with Update to persist.
Update	Empty	Persists the current row to the database (after AddNew or field modifications).
Delete	Empty	Deletes the current row.
GetRows	Array or Empty	Returns a 2D array of the entire result set (columns

Method	Returns	Description
		first, then rows). Empty if Recordset is closed or empty.
GetString	String	Returns the entire result set as a tab-separated string with newline row delimiters.
Requery	Empty	Re-executes the original query to refresh the Recordset.
Resync	Empty	Refreshes the current row data from the database (no-op in current runtime).
Save	Empty	Writes the Recordset to a file or ADODB.Stream object in XML format.
Seek	Empty	Positions the current row by searching the index. Requires the current Recordset to have an Index property set.
Supports	Boolean	Returns true if the Recordset supports the specified feature (e.g., bookmarks, seeking).
UpdateBatch	Empty	Applies all pending batch updates to the database (no-op in current runtime).

447.3 Command Methods

Method	Returns	Description
Execute	Recordset, Integer, or Empty	Executes the SQL statement in CommandText. Returns a Recordset for queries, an Integer row count for modifications, or Empty on failure. Use

Method	Returns	Description
		Connection.Execute for simplified direct execution.
CreateParameter	Parameter	Creates a Parameter object with the specified data type, direction (in/out), size, and value.
Cancel	Empty	Cancels the current operation (no-op in current runtime).

447.4 Collection Methods

Object	Method	Returns	Description
Errors	Count	Integer	Returns the number of errors in the collection.
Errors	Item	Error or Empty	Returns the error at the specified index, or Empty if out of range.
Errors	Clear	Empty	Clears all errors from the collection.
Fields	Append	Empty	Adds a new Field definition to the collection (schema design).
Fields	Item	Field	Returns the Field at the specified index or by name.
Fields	Count	Integer	Returns the number of fields in the collection.
Field	AppendChunk	Empty	Appends binary or text data to a Blob field.
Field	GetChunk	String	Retrieves a chunk of binary or text data from a Blob

Object	Method	Returns	Description
			field.
Parameters	Append	Empty	Adds a Parameter to the command's parameter collection.
Parameters	Count	Integer	Returns the number of parameters in the collection.

447.5 Code Example

```
<%
Option Explicit
Dim conn, rs, parms, affected

Set conn = Server.CreateObject("ADODB.Connection")
conn.ConnectionString = "Driver=sqlite;Data Source=" & Server.MapPath("./
data/app.db")
conn.Open

affected = conn.Execute("UPDATE users SET active = 1 WHERE active = 0")
Response.Write "Rows updated: " & CStr(affected) & "<br>"

Set rs = conn.Execute("SELECT id, name FROM users")
Do While Not rs.EOF
    Response.Write rs.Fields("name").Value & "<br>"
    rs.MoveNext
Loop

rs.Close
conn.Close
Set rs = Nothing
Set conn = Nothing
%>
```

448 ADODB Properties Reference

448.1 Connection Properties

Property	Access	Type	Description
ConnectionString	Read/Write	String	Full provider connection string (e.g., `Driver={SQLite3};Data`

Property	Access	Type	Description
			Source=app.db).
State	Read	Integer	Connection state: 0 (closed), 1 (open).
Mode	Read/Write	Integer	Access mode for the connection.
Provider	Read/Write	String	Provider name (e.g., sqloledb, msdasql).
Version	Read	String	Version string for compatibility reporting.
CommandTimeout	Read/Write	Integer	Default timeout in seconds for command execution.
ConnectionTimeout	Read/Write	Integer	Timeout in seconds for establishing the connection.
CursorLocation	Read/Write	Integer	Cursor location: 1 (server-side), 3 (client-side).
DefaultDatabase	Read/Write	String	Default database or catalog name.
IsolationLevel	Read/Write	Integer	Transaction isolation level.
Errors	Read	Object	Errors collection containing all errors from the last failed operation.

448.2 Recordset Properties

Property	Access	Type	Description
EOF	Read	Boolean	True when cursor is positioned after the last row.
BOF	Read	Boolean	True when cursor is positioned before

Property	Access	Type	Description
			the first row.
State	Read	Integer	Recordset state: 0 (closed), 1 (open).
RecordCount	Read	Integer	Number of rows currently materialized in memory.
Fields	Read	Object	Fields collection representing the current column schema.
AbsolutePage	Read/Write	Integer	Current page index (used with PageSize).
PageCount	Read	Integer	Total page count based on RecordCount and PageSize.
PageSize	Read/Write	Integer	Number of rows per page.
AbsolutePosition	Read/Write	Integer	One-based position of the current row.
Bookmark	Read/Write	Variant	Bookmark value for the current row (use with Seek).
ActiveCommand	Read/Write	Object	Associated Command object that produced this Recordset.
CacheSize	Read/Write	Integer	Number of rows to cache locally (minimum effective value is 1).
DataMember	Read/Write	String	Data member name for complex data sources.
EditMode	Read	Integer	Current edit mode: 0 (not editing), 1

Property	Access	Type	Description
			(edit), 2 (add new).
Filter	Read/Write	String	Filter clause to restrict visible rows (format: `field = value`).
Index	Read/Write	String	Indexed field expression for Seek operations.
MarshalOptions	Read/Write	Integer	Options for marshaling data.
MaxRecords	Read/Write	Integer	Maximum number of rows to materialize from the source.
Sort	Read/Write	String	Sort order for rows (format: `field ASC` or `field DESC`).
Source	Read/Write	String or Object	SQL source text or Command object to execute.
Status	Read/Write	Integer	Status of the current record.
ActiveConnection	Write	Object	Connection object used by Open and persistence methods.
CursorType	Write	Integer	Cursor type: 0 (forward), 1 (keyset), 2 (dynamic), 3 (static).
LockType	Write	Integer	Lock type: 1 (read-only), 2 (pessimistic), 3 (optimistic), 4 (batch).
CursorLocation	Write	Integer	Overrides connection cursor

Property	Access	Type	Description
			location for this Recordset.

448.3 Command Properties

Property	Access	Type	Description
ActiveConnection	Read/Write	Object	Connection object used for Execute.
CommandText	Read/Write	String	SQL statement or stored procedure name.
CommandType	Read/Write	Integer	Command type: 1 (text), 2 (table), 4 (stored procedure).
CommandTimeout	Read/Write	Integer	Timeout in seconds for command execution.
Prepared	Read/Write	Boolean	Whether the command should be prepared before execution.
Parameters	Read	Object	Parameters collection for the command.

448.4 Error Properties

Property	Access	Type	Description
Number	Read	Integer	HRESULT-compatible error number.
Description	Read	String	Human-readable error message.
Source	Read	String	Member name or function that raised the error.
SQLState	Read	String	SQL state code when available.

448.5 Errors Collection Properties

Property	Access	Type	Description
Count	Read	Integer	Number of errors in the collection.
Item	Read	Object	Access individual Error objects by index.

448.6 Field Properties

Property	Access	Type	Description
Value	Read/Write	Variant	Current field value.
Name	Read	String	Field name from the query schema.
Type	Read	Integer	Field data type code.
DefinedSize	Read	Integer	Declared maximum size of the field.
Attributes	Read	Integer	Field attribute flags (nullable, auto-increment, etc.).
NumericScale	Read/Write	Integer	Number of decimal places for numeric fields.
ActualSize	Read	Integer	Actual length in characters of the current value.
DataFormat	Read	Empty	Returns Empty in current runtime.
OriginalValue	Read	Variant	Original value before modification (mirrors current value in current runtime).
Precision	Read	Integer	Number of significant digits for numeric fields.
Status	Read	Integer	Field status (inherited from Recordset status).
UnderlyingValue	Read	Variant	Database value before local

Property	Access	Type	Description
			modifications (mirrors current value in current runtime).

448.7 Fields Collection Properties

Property	Access	Type	Description
Count	Read	Integer	Number of fields in the Recordset schema.
Item	Read	Object	Access individual Field objects by index or name.

448.8 Parameter Properties

Property	Access	Type	Description
Name	Read/Write	String	Parameter name (used in parameterized queries).
Value	Read/Write	Variant	Parameter value to pass to the SQL command.
Type	Read/Write	Integer	Parameter data type.
Direction	Read/Write	Integer	Parameter direction: 1 (input), 2 (output), 4 (input-output).
Size	Read/Write	Integer	Maximum parameter size in bytes.

448.9 Parameters Collection Properties

Property	Access	Type	Description
Count	Read	Integer	Number of parameters in the collection.

Property	Access	Type	Description
Item	Read	Object	Access individual Parameter objects by index or name.

449 Connection.Open Method

Opens the database connection using the configured connection string.

449.1 Syntax

```
conn.Open [connectionString]
```

449.2 Parameters

Parameter	Type	Required	Description
`connectionString`	String	No	Overrides `ConnectionString` for this call.

449.3 Return Value

Empty. The method does not return a value.

449.4 Remarks

- Method names are case-insensitive.
- If no parameter is provided, `ConnectionString` must already be set.
- After a successful call, `State` becomes `1`.

449.5 Code Example

```
<%
Option Explicit
Dim conn

Set conn = Server.CreateObject("ADODB.Connection")
conn.ConnectionString = "Driver={SQLite3};Data Source=" & Server.MapPath
("./db.sqlite")
conn.Open

Response.Write "State: " & CStr(conn.State)

conn.Close
Set conn = Nothing
%>
```

450 Connection.Close Method

Closes an open connection and releases provider resources.

450.1 Syntax

```
conn.Close
```

450.2 Parameters

No parameters.

450.3 Return Value

Empty. The method does not return a value.

450.4 Remarks

- Method names are case-insensitive.
- Call this method when all recordsets are closed.
- After closing, `State` becomes `0`.

450.5 Code Example

```
<%
Option Explicit
Dim conn

Set conn = Server.CreateObject("ADODB.Connection")
conn.ConnectionString = "Driver={SQLite3};Data Source=" & Server.MapPath
("./db.sqlite")
conn.Open

Response.Write "Opened<br>"
conn.Close
Response.Write "Closed"

Set conn = Nothing
%>
```

451 Connection.Execute Method

Executes a SQL statement directly on the connection.

451.1 Syntax

```
result = conn.Execute(commandText)
```

451.2 Parameters

Parameter	Type	Required	Description
`commandText`	String	Yes	SQL statement to run.

451.3 Return Value

Variant. Returns an ADODB.Recordset for query statements. Returns an Integer for affected rows in action statements. Returns Empty on failure.

451.4 Remarks

- Method names are case-insensitive.
- Query detection is based on SQL statement type in the runtime.
- Use this method for quick execution when a dedicated Command object is not needed.

451.5 Code Example

```
<%
Option Explicit
Dim conn, result

Set conn = Server.CreateObject("ADODB.Connection")
conn.ConnectionString = "Driver={SQLite3};Data Source=" & Server.MapPath
("./db.sqlite")
conn.Open

Set result = conn.Execute("SELECT id, name FROM users")
If Not result.EOF Then
    Response.Write result.Fields("name").Value
End If

result.Close
conn.Close
Set result = Nothing
Set conn = Nothing
%>
```

452 Connection.BeginTrans Method

Starts a transaction on the current connection.

452.1 Syntax

```
level = conn.BeginTrans
```

452.2 Parameters

No parameters.

452.3 Return Value

Integer. Returns `1` when the transaction starts successfully; returns `0` otherwise.

452.4 Remarks

- Method names are case-insensitive.
- Pair this method with `CommitTrans` or `RollbackTrans`.
- Use transactions when multiple statements must succeed or fail together.

452.5 Code Example

```
<%
Option Explicit
Dim conn, level

Set conn = Server.CreateObject("ADODB.Connection")
conn.ConnectionString = "Driver={SQLite3};Data Source=" & Server.MapPath
("./db.sqlite")
conn.Open

level = conn.BeginTrans
If level = 1 Then
    conn.Execute "UPDATE users SET active = 1 WHERE id = 1"
    conn.CommitTrans
    Response.Write "Committed"
Else
    Response.Write "Transaction not started"
End If

conn.Close
Set conn = Nothing
%>
```

453 Connection.CommitTrans Method

Commits the current transaction and persists all pending changes.

453.1 Syntax

```
conn.CommitTrans
```

453.2 Parameters

No parameters.

453.3 Return Value

Empty. The method does not return a value.

453.4 Remarks

- Method names are case-insensitive.
- Call this method after a successful `BeginTrans` block.
- After commit, changes cannot be rolled back.

453.5 Code Example

```
<%
Option Explicit
Dim conn

Set conn = Server.CreateObject("ADODB.Connection")
conn.ConnectionString = "Driver={SQLite3};Data Source=" & Server.MapPath
("./db.sqlite")
conn.Open

conn.BeginTrans
conn.Execute "UPDATE users SET active = 1 WHERE id = 1"
conn.CommitTrans

Response.Write "Transaction committed"

conn.Close
Set conn = Nothing
%>
```

454 Connection.RollbackTrans Method

Rolls back the active transaction and discards pending changes.

454.1 Syntax

```
conn.RollbackTrans
```

454.2 Parameters

No parameters.

454.3 Return Value

Empty. The method does not return a value.

454.4 Remarks

- Method names are case-insensitive.

- Call this method after `BeginTrans` when execution fails.
- Rollback restores the state from before transaction start.

454.5 Code Example

```
<%
Option Explicit
Dim conn

Set conn = Server.CreateObject("ADODB.Connection")
conn.ConnectionString = "Driver={SQLite3};Data Source=" & Server.MapPath
("./db.sqlite")
conn.Open

conn.BeginTrans
On Error Resume Next
conn.Execute "UPDATE users SET active = 0 WHERE id = 1"
If Err.Number <> 0 Then
    conn.RollbackTrans
    Response.Write "Transaction rolled back"
Else
    conn.CommitTrans
End If
On Error GoTo 0

conn.Close
Set conn = Nothing
%>
```

455 Connection.OpenSchema Method

Returns schema metadata as a Recordset.

455.1 Syntax

```
Set rs = conn.OpenSchema(schemaId[, restrictions])
```

455.2 Parameters

Parameter	Type	Required	Description
`schemaId`	Integer	No	Schema type selector.
`restrictions`	Variant	No	Restriction array or provider-specific filter values.

455.3 Return Value

Recordset. Returns an ADODB.Recordset containing schema rows for the requested schema type.

455.4 Remarks

- Method names are case-insensitive.
- Supported schema sets depend on provider capabilities.
- Validate field availability in returned schema recordset.

455.5 Code Example

```
<%
Option Explicit
Dim conn, rs

Set conn = Server.CreateObject("ADODB.Connection")
conn.ConnectionString = "Driver={SQLite3};Data Source=" & Server.MapPath
("./db.sqlite")
conn.Open

Set rs = conn.OpenSchema(20)
If Not rs.EOF Then
    Response.Write "Schema row available"
End If

rs.Close
conn.Close
Set rs = Nothing
Set conn = Nothing
%>
```

456 Connection.Cancel Method

Cancels the current connection operation.

456.1 Syntax

```
conn.Cancel
```

456.2 Parameters

No parameters.

456.3 Return Value

Empty. The method does not return a value.

456.4 Remarks

- Method names are case-insensitive.
- In the current G3Pix AxonASP ADODB implementation, this method is a compatibility no-op.
- Use this method only when porting legacy scripts that call Connection.Cancel.
- The method does not close the connection.

456.5 Code Example

```
<%
Option Explicit
Dim conn

Set conn = Server.CreateObject("ADODB.Connection")
conn.ConnectionString = "Driver={SQLite3};Data Source=" & Server.MapPath
("./db.sqlite")
conn.Open

' Compatibility no-op in current runtime.
conn.Cancel

Response.Write "Connection.Cancel executed" & "<br>"
Response.Write "Connection state: " & CStr(conn.State)

conn.Close
Set conn = Nothing
%>
```

457 Recordset.Open Method

Opens a recordset using a source query and connection.

457.1 Syntax

```
rs.Open source, activeConnection[, cursorType[, lockType[, options]]]
```

457.2 Parameters

Parameter	Type	Required	Description
`source`	String or Command	Yes	SQL text, table name, or command object.
`activeConnection`	Connection	Yes	Open connection used to fetch rows.
`cursorType`	Integer	No	Cursor behavior flag.

Parameter	Type	Required	Description
`lockType`	Integer	No	Lock mode flag.
`options`	Integer	No	Provider options bitmask.

457.3 Return Value

Empty. The method does not return a value.

457.4 Remarks

- Method names are case-insensitive.
- After open, use BOF/EOF and Fields for navigation and values.
- Always close the recordset after use.

457.5 Code Example

```
<%
Option Explicit
Dim conn, rs

Set conn = Server.CreateObject("ADODB.Connection")
conn.ConnectionString = "Driver={SQLite3};Data Source=" & Server.MapPath
("./db.sqlite")
conn.Open

Set rs = Server.CreateObject("ADODB.Recordset")
rs.Open "SELECT id, name FROM users", conn

If Not rs.EOF Then Response.Write rs.Fields("name").Value

rs.Close
conn.Close
Set rs = Nothing
Set conn = Nothing
%>
```

458 Recordset.Close Method

Closes the recordset and releases cursor resources.

458.1 Syntax

```
rs.Close
```

458.2 Parameters

No parameters.

458.3 Return Value

Empty. The method does not return a value.

458.4 Remarks

- Method names are case-insensitive.
- Call this method before closing the parent connection.
- After close, `State` becomes `0`.

458.5 Code Example

```
<%
Option Explicit
Dim conn, rs

Set conn = Server.CreateObject("ADODB.Connection")
conn.ConnectionString = "Driver={SQLite3};Data Source=" & Server.MapPath
("./db.sqlite")
conn.Open

Set rs = conn.Execute("SELECT id FROM users")
rs.Close

Response.Write "Recordset state: " & CStr(rs.State)

conn.Close
Set rs = Nothing
Set conn = Nothing
%>
```

459 Recordset.MoveNext Method

Moves the cursor to the next row.

459.1 Syntax

```
rs.MoveNext
```

459.2 Parameters

No parameters.

459.3 Return Value

Empty. The method does not return a value.

459.4 Remarks

- Method names are case-insensitive.

- EOF becomes True when the cursor moves past the last row.
- Use this in forward iteration loops.

459.5 Code Example

```
<%
Option Explicit
Dim conn, rs

Set conn = Server.CreateObject("ADODB.Connection")
conn.ConnectionString = "Driver={SQLite3};Data Source=" & Server.MapPath
("./db.sqlite")
conn.Open
Set rs = conn.Execute("SELECT id, name FROM users")

Do While Not rs.EOF
    Response.Write rs.Fields("name").Value & "<br>"
    rs.MoveNext
Loop

rs.Close
conn.Close
Set rs = Nothing
Set conn = Nothing
%>
```

460 Recordset.MovePrevious Method

Moves the cursor to the previous row in the result set.

460.1 Syntax

```
rs.MovePrevious
```

460.2 Parameters

No parameters.

460.3 Return Value

Empty. The method does not return a value.

460.4 Remarks

- Method names are case-insensitive.
- After calling MovePrevious, check BOF to verify whether the cursor moved before the first row.
- Calling MovePrevious on the first row sets BOF to True.
- Use MovePrevious only with cursor types that support backward navigation.

460.5 Code Example

```
<%
Option Explicit
Dim conn, rs

Set conn = Server.CreateObject("ADODB.Connection")
conn.ConnectionString = "Driver={SQLite3};Data Source=" & Server.MapPath
("./db.sqlite")
conn.Open

Set rs = conn.Execute("SELECT id, name FROM users")
rs.MoveLast
Response.Write "Last row: " & rs.Fields("name").Value & "<br>"

rs.MovePrevious
If Not rs.BOF Then
    Response.Write "Previous row: " & rs.Fields("name").Value
End If

rs.Close
conn.Close
Set rs = Nothing
Set conn = Nothing
%>
```

461 Recordset.MoveFirst Method

Moves the cursor to the first row.

461.1 Syntax

```
rs.MoveFirst
```

461.2 Parameters

No parameters.

461.3 Return Value

Empty. The method does not return a value.

461.4 Remarks

- Method names are case-insensitive.
- Use this method to reset iteration to row 1.
- BOF becomes False when positioned on a valid first row.

461.5 Code Example

```
<%
Option Explicit
Dim conn, rs

Set conn = Server.CreateObject("ADODB.Connection")
conn.ConnectionString = "Driver={SQLite3};Data Source=" & Server.MapPath
("./db.sqlite")
conn.Open

Set rs = conn.Execute("SELECT id, name FROM users")
rs.MoveLast
rs.MoveFirst

If Not rs.EOF Then Response.Write rs.Fields("name").Value

rs.Close
conn.Close
Set rs = Nothing
Set conn = Nothing
%>
```

462 Recordset.MoveLast Method

Moves the cursor to the last row.

462.1 Syntax

```
rs.MoveLast
```

462.2 Parameters

No parameters.

462.3 Return Value

Empty. The method does not return a value.

462.4 Remarks

- Method names are case-insensitive.
- Use this method when you need the final row or to force full row materialization.
- After MoveLast, EOF is False on non-empty rowsets.

462.5 Code Example

```
<%
Option Explicit
```

```
Dim conn, rs

Set conn = Server.CreateObject("ADODB.Connection")
conn.ConnectionString = "Driver={SQLite3};Data Source=" & Server.MapPath
("./db.sqlite")
conn.Open

Set rs = conn.Execute("SELECT id, name FROM users")
rs.MoveLast

If Not rs.EOF Then Response.Write rs.Fields("name").Value

rs.Close
conn.Close
Set rs = Nothing
Set conn = Nothing
%>
```

463 Recordset.Move Method

Moves the current cursor position by a relative row offset.

463.1 Syntax

```
rs.Move offset
```

463.2 Parameters

Parameter	Type	Required	Description
`offset`	Integer	Yes	Positive to move forward, negative to move backward.

463.3 Return Value

Empty. The method does not return a value.

463.4 Remarks

- Method names are case-insensitive.
- Cursor movement beyond bounds sets BOF or EOF accordingly.
- Move supports flexible navigation without repeated MoveNext or MovePrevious calls.
- Use with cursor types that support requested direction.

463.5 Code Example

```
<%
Option Explicit
```

```
Dim conn, rs

Set conn = Server.CreateObject("ADODB.Connection")
conn.ConnectionString = "Driver={SQLite3};Data Source=" & Server.MapPath
("./db.sqlite")
conn.Open

Set rs = conn.Execute("SELECT id, name FROM users")
rs.MoveFirst
rs.Move 2

If Not rs.EOF Then
    Response.Write "Current row after Move 2: " & rs.Fields("name").Value
End If

rs.Close
conn.Close
Set rs = Nothing
Set conn = Nothing
%>
```

464 Recordset.Cancel Method

Cancels the current recordset operation.

464.1 Syntax

```
rs.Cancel
```

464.2 Parameters

No parameters.

464.3 Return Value

Empty. The method does not return a value.

464.4 Remarks

- Method names are case-insensitive.
- In the current G3Pix AxonASP ADODB implementation, this method is a compatibility no-op.
- Use this method only to preserve compatibility with legacy scripts.
- It does not close the recordset or revert changes.

464.5 Code Example

```
<%
Option Explicit
```

```
Dim conn, rs

Set conn = Server.CreateObject("ADODB.Connection")
conn.ConnectionString = "Driver={SQLite3};Data Source=" & Server.MapPath
("./db.sqlite")
conn.Open

Set rs = conn.Execute("SELECT id, name FROM users")

' Compatibility no-op in current runtime.
rs.Cancel

Response.Write "Recordset.Cancel executed"

rs.Close
conn.Close
Set rs = Nothing
Set conn = Nothing
%>
```

465 Recordset.CancelBatch Method

Cancels pending batch updates in a batch-update workflow.

465.1 Syntax

```
rs.CancelBatch
```

465.2 Parameters

No parameters.

465.3 Return Value

Empty. The method does not return a value.

465.4 Remarks

- Method names are case-insensitive.
- In the current G3Pix AxonASP ADODB implementation, this method is a compatibility no-op.
- Use this method for script compatibility when batch mode is not required.
- For current behavior, pending edits are managed with AddNew/Update/CancelUpdate.

465.5 Code Example

```
<%
Option Explicit
```

```
Dim conn, rs

Set conn = Server.CreateObject("ADODB.Connection")
conn.ConnectionString = "Driver={SQLite3};Data Source=" & Server.MapPath
("./db.sqlite")
conn.Open

Set rs = conn.Execute("SELECT * FROM users")

' Compatibility no-op in current runtime.
rs.CancelBatch

Response.Write "Recordset.CancelBatch executed"

rs.Close
conn.Close
Set rs = Nothing
Set conn = Nothing
%>
```

466 Recordset.CancelUpdate Method

Cancels pending edits on the current row before they are persisted.

466.1 Syntax

```
rs.CancelUpdate
```

466.2 Parameters

No parameters.

466.3 Return Value

Empty. The method does not return a value.

466.4 Remarks

- Method names are case-insensitive.
- Use CancelUpdate after modifying fields or calling AddNew when you want to discard changes.
- CancelUpdate affects only the current pending row change.
- After CancelUpdate, field values revert to the previously committed state.

466.5 Code Example

```
<%
Option Explicit
Dim conn, rs
```

```
Set conn = Server.CreateObject("ADODB.Connection")
conn.ConnectionString = "Driver={SQLite3};Data Source=" & Server.MapPath
("./db.sqlite")
conn.Open

Set rs = conn.Execute("SELECT * FROM users WHERE id = 1")
If Not rs.EOF Then
    rs.Fields("name").Value = "Temporary Name"
    rs.CancelUpdate
    Response.Write "Update canceled"
End If

rs.Close
conn.Close
Set rs = Nothing
Set conn = Nothing
%>
```

467 Recordset.Find Method

Searches the recordset for the next row that matches a criteria expression.

467.1 Syntax

```
rs.Find criteria
```

467.2 Parameters

Parameter	Type	Required	Description
`criteria`	String	Yes	Match expression, such as `"id = 10"` or `"name = 'Ana'"`.

467.3 Return Value

Empty. The method does not return a value.

467.4 Remarks

- Method names are case-insensitive.
- Find moves the current cursor position to the first matching row at or after the current position.
- If no row matches, EOF becomes True.
- Criteria parsing supports the runtime-compatible subset used by AxonASP ADODB.

467.5 Code Example

```
<%
Option Explicit
Dim conn, rs

Set conn = Server.CreateObject("ADODB.Connection")
conn.ConnectionString = "Driver={SQLite3};Data Source=" & Server.MapPath
("./db.sqlite")
conn.Open

Set rs = conn.Execute("SELECT id, name FROM users")
rs.Find "id = 1"

If Not rs.EOF Then
    Response.Write "Found: " & rs.Fields("name").Value
Else
    Response.Write "No match"
End If

rs.Close
conn.Close
Set rs = Nothing
Set conn = Nothing
%>
```

468 Recordset.NextRecordset Method

Returns the next recordset when a command produces multiple result sets.

468.1 Syntax

```
Set nextRs = rs.NextRecordset
```

468.2 Parameters

No parameters.

468.3 Return Value

Recordset or Nothing. Returns the next ADODB.Recordset object if available; returns Nothing when there are no additional result sets.

468.4 Remarks

- Method names are case-insensitive.
- Use NextRecordset only when SQL/provider can emit multiple result sets.
- Always check if returned object is Nothing before using it.
- Close each returned recordset when finished.

468.5 Code Example

```
<%
Option Explicit
Dim conn, rs, nextRs

Set conn = Server.CreateObject("ADODB.Connection")
conn.ConnectionString = "Driver={SQLite3};Data Source=" & Server.MapPath
("./db.sqlite")
conn.Open

Set rs = conn.Execute("SELECT id FROM users")
Set nextRs = rs.NextRecordset

If IsObject(nextRs) Then
    Response.Write "Second recordset available"
    nextRs.Close
Else
    Response.Write "No additional recordset"
End If

rs.Close
conn.Close
Set nextRs = Nothing
Set rs = Nothing
Set conn = Nothing
%>
```

469 Recordset.Clone Method

Creates a new Recordset object that references the same rowset data.

469.1 Syntax

```
Set rsClone = rs.Clone
```

469.2 Parameters

No parameters.

469.3 Return Value

Recordset. Returns a new ADODB.Recordset object clone.

469.4 Remarks

- Method names are case-insensitive.
- The clone has independent cursor position.

- Changes to shared underlying data can be visible in both recordsets, depending on lock and cursor behavior.
- Close and release cloned recordsets separately.

469.5 Code Example

```
<%
Option Explicit
Dim conn, rs, rsClone

Set conn = Server.CreateObject("ADODB.Connection")
conn.ConnectionString = "Driver={SQLite3};Data Source=" & Server.MapPath
("./db.sqlite")
conn.Open

Set rs = conn.Execute("SELECT id, name FROM users")
Set rsClone = rs.Clone

If Not rs.EOF Then
    Response.Write "Original current row: " & rs.Fields("name").Value & "
<br>"
End If

If Not rsClone.EOF Then
    Response.Write "Clone current row: " & rsClone.Fields("name").Value
End If

rsClone.Close
rs.Close
conn.Close
Set rsClone = Nothing
Set rs = Nothing
Set conn = Nothing
%>
```

470 Recordset.CompareBookmarks Method

Compares two bookmark positions from the same recordset.

470.1 Syntax

```
comparison = rs.CompareBookmarks(bookmark1, bookmark2)
```

470.2 Parameters

Parameter	Type	Required	Description
bookmark1	Variant	Yes	First bookmark value.
bookmark2	Variant	Yes	Second bookmark value.

470.3 Return Value

Integer. Returns `-1` when `bookmark1` is before `bookmark2`, `0` when both point to the same row, and `1` when `bookmark1` is after `bookmark2`.

470.4 Remarks

- Method names are case-insensitive.
- Both bookmarks must come from the same recordset cursor context.
- Invalid bookmarks raise runtime errors.
- Use this method for cursor position logic without moving the active row.

470.5 Code Example

```
<%
Option Explicit
Dim conn, rs, bm1, bm2, cmp

Set conn = Server.CreateObject("ADODB.Connection")
conn.ConnectionString = "Driver={SQLite3};Data Source=" & Server.MapPath
("./db.sqlite")
conn.Open

Set rs = conn.Execute("SELECT id, name FROM users")
If Not rs.EOF Then
    bm1 = rs.Bookmark
    rs.MoveNext
    If Not rs.EOF Then
        bm2 = rs.Bookmark
        cmp = rs.CompareBookmarks(bm1, bm2)
        Response.Write "Compare result: " & CStr(cmp)
    End If
End If

rs.Close
conn.Close
Set rs = Nothing
Set conn = Nothing
%>
```

471 Recordset.AddNew Method

Creates a new pending row in the current recordset.

471.1 Syntax

```
rs.AddNew
```

471.2 Parameters

No parameters.

471.3 Return Value

Empty. The method does not return a value.

471.4 Remarks

- Method names are case-insensitive.
- Set field values after AddNew and call `Update` to persist.
- Use `CancelUpdate` to discard pending row changes.

471.5 Code Example

```
<%
Option Explicit
Dim conn, rs

Set conn = Server.CreateObject("ADODB.Connection")
conn.ConnectionString = "Driver={SQLite3};Data Source=" & Server.MapPath
("./db.sqlite")
conn.Open

Set rs = conn.Execute("SELECT * FROM users")
rs.AddNew
rs.Fields("name").Value = "New User"
rs.Update

Response.Write "Row added"

rs.Close
conn.Close
Set rs = Nothing
Set conn = Nothing
%>
```

472 Recordset.Update Method

Persists pending edits for the current row.

472.1 Syntax

```
rs.Update
```

472.2 Parameters

No parameters.

472.3 Return Value

Empty. The method does not return a value.

472.4 Remarks

- Method names are case-insensitive.
- Use after `AddNew` or field value changes.
- Requires an updatable recordset and valid connection context.

472.5 Code Example

```
<%
Option Explicit
Dim conn, rs

Set conn = Server.CreateObject("ADODB.Connection")
conn.ConnectionString = "Driver={SQLite3};Data Source=" & Server.MapPath
("./db.sqlite")
conn.Open
Set rs = conn.Execute("SELECT * FROM users WHERE id = 1")

If Not rs.EOF Then
    rs.Fields("name").Value = "Updated Name"
    rs.Update
End If

rs.Close
conn.Close
Set rs = Nothing
Set conn = Nothing
%>
```

473 Recordset.Delete Method

Deletes the current row from the recordset and data source.

473.1 Syntax

```
rs.Delete
```

473.2 Parameters

No parameters.

473.3 Return Value

Empty. The method does not return a value.

473.4 Remarks

- Method names are case-insensitive.
- Deletion applies to the current row position.
- Move to the next valid row after deletion when iterating.

473.5 Code Example

```
<%
Option Explicit
Dim conn, rs

Set conn = Server.CreateObject("ADODB.Connection")
conn.ConnectionString = "Driver={SQLite3};Data Source=" & Server.MapPath
("./db.sqlite")
conn.Open

Set rs = conn.Execute("SELECT * FROM users WHERE id = 1")
If Not rs.EOF Then
    rs.Delete
    Response.Write "Row deleted"
End If

rs.Close
conn.Close
Set rs = Nothing
Set conn = Nothing
%>
```

474 Recordset.GetRows Method

Returns recordset rows as a two-dimensional array.

474.1 Syntax

```
rows = rs.GetRows
```

474.2 Parameters

No parameters.

474.3 Return Value

Array or Empty. Returns a 2D array in `[columnIndex][rowIndex]` order. Returns Empty when the recordset is closed or has no rows.

474.4 Remarks

- Method names are case-insensitive.
- This method reads the full remaining rowset into memory.

- Use for bulk processing when array access is preferred.

474.5 Code Example

```
<%
Option Explicit
Dim conn, rs, rows, i

Set conn = Server.CreateObject("ADODB.Connection")
conn.ConnectionString = "Driver={SQLite3};Data Source=" & Server.MapPath
("./db.sqlite")
conn.Open

Set rs = conn.Execute("SELECT id, name FROM users")
rows = rs.GetRows

If IsArray(rows) Then
    For i = 0 To UBound(rows, 2)
        Response.Write rows(0, i) & " - " & rows(1, i) & "<br>"
    Next
End If

rs.Close
conn.Close
Set rs = Nothing
Set conn = Nothing
%>
```

475 Recordset.GetString Method

Returns recordset rows as a single formatted string.

475.1 Syntax

```
text = rs.GetString
```

475.2 Parameters

No parameters.

475.3 Return Value

String. Returns row data with tab-delimited columns and newline-delimited rows. Returns an empty string when no rows are available.

475.4 Remarks

- Method names are case-insensitive.
- Output is useful for export, debug logs, or plain-text rendering.

- Large datasets may produce large memory allocations.

475.5 Code Example

```
<%
Option Explicit
Dim conn, rs, text

Set conn = Server.CreateObject("ADODB.Connection")
conn.ConnectionString = "Driver={SQLite3};Data Source=" & Server.MapPath
("./db.sqlite")
conn.Open

Set rs = conn.Execute("SELECT id, name FROM users")
text = rs.GetString

Response.Write "<pre>" & text & "</pre>"

rs.Close
conn.Close
Set rs = Nothing
Set conn = Nothing
%>
```

476 Recordset.Requery Method

Re-executes the original query and refreshes the recordset rows.

476.1 Syntax

```
rs.Requery
```

476.2 Parameters

No parameters.

476.3 Return Value

Empty. The method does not return a value.

476.4 Remarks

- Method names are case-insensitive.
- Requery uses the recordset source and active connection currently associated with the recordset.
- Cursor position can change after refresh.
- Use Requery when underlying table data may have changed.

476.5 Code Example

```
<%
Option Explicit
Dim conn, rs

Set conn = Server.CreateObject("ADODB.Connection")
conn.ConnectionString = "Driver={SQLite3};Data Source=" & Server.MapPath
("./db.sqlite")
conn.Open

Set rs = Server.CreateObject("ADODB.Recordset")
rs.Open "SELECT id, name FROM users", conn

Response.Write "Before requery count: " & CStr(rs.RecordCount) & "<br>"
rs.Requery
Response.Write "After requery count: " & CStr(rs.RecordCount)

rs.Close
conn.Close
Set rs = Nothing
Set conn = Nothing
%>
```

477 Recordset.Resync Method

Refreshes the current row values from the data source.

477.1 Syntax

```
rs.Resync
```

477.2 Parameters

No parameters.

477.3 Return Value

Empty. The method does not return a value.

477.4 Remarks

- Method names are case-insensitive.
- In the current G3Pix AxonASP ADODB implementation, this method is a compatibility no-op.
- Use it to keep compatibility with legacy scripts that call Resync.

477.5 Code Example

```
<%
Option Explicit
Dim conn, rs

Set conn = Server.CreateObject("ADODB.Connection")
conn.ConnectionString = "Driver={SQLite3};Data Source=" & Server.MapPath
("./db.sqlite")
conn.Open

Set rs = conn.Execute("SELECT id, name FROM users")
rs.Resync

Response.Write "Resync executed"

rs.Close
conn.Close
Set rs = Nothing
Set conn = Nothing
%>
```

478 Recordset.Save Method

Saves recordset content to a file path or stream target.

478.1 Syntax

```
rs.Save destination
```

478.2 Parameters

Parameter	Type	Required	Description
`destination`	String or Object	Yes	Output file path or stream object target.

478.3 Return Value

Empty. The method does not return a value.

478.4 Remarks

- Method names are case-insensitive.
- Output format depends on runtime implementation and destination handling.
- Ensure destination path is writable.

478.5 Code Example

```
<%
Option Explicit
Dim conn, rs, path

Set conn = Server.CreateObject("ADODB.Connection")
conn.ConnectionString = "Driver={SQLite3};Data Source=" & Server.MapPath
("./db.sqlite")
conn.Open
Set rs = conn.Execute("SELECT id, name FROM users")

path = Server.MapPath("./users.xml")
rs.Save path
Response.Write "Saved to: " & path

rs.Close
conn.Close
Set rs = Nothing
Set conn = Nothing
%>
```

479 Recordset.Seek Method

Repositions the cursor using an index key.

479.1 Syntax

```
rs.Seek keyOrPosition
```

479.2 Parameters

Parameter	Type	Required	Description
`keyOrPosition`	Variant	Yes	Key or position value used by the provider seek implementation.

479.3 Return Value

Empty. The method does not return a value.

479.4 Remarks

- Method names are case-insensitive.
- Seek support depends on provider and recordset index configuration.
- Set `Recordset.Index` before calling Seek when required.

479.5 Code Example

```
<%
Option Explicit
Dim conn, rs

Set conn = Server.CreateObject("ADODB.Connection")
conn.ConnectionString = "Driver={SQLite3};Data Source=" & Server.MapPath
("./db.sqlite")
conn.Open
Set rs = conn.Execute("SELECT id, name FROM users")

rs.Index = "id"
rs.Seek 1
If Not rs.EOF Then Response.Write rs.Fields("name").Value

rs.Close
conn.Close
Set rs = Nothing
Set conn = Nothing
%>
```

480 Recordset.Supports Method

Checks whether a recordset feature flag is supported.

480.1 Syntax

```
ok = rs.Supports(feature)
```

480.2 Parameters

Parameter	Type	Required	Description
`feature`	Integer	Yes	Feature capability constant to test.

480.3 Return Value

Boolean. Returns True when the feature is supported; otherwise False.

480.4 Remarks

- Method names are case-insensitive.
- Use this before calling optional operations like bookmarks or batch updates.
- Supported features vary by provider and cursor configuration.

480.5 Code Example

```
<%
Option Explicit
Dim conn, rs, ok
```

```
Set conn = Server.CreateObject("ADODB.Connection")
conn.ConnectionString = "Driver={SQLite3};Data Source=" & Server.MapPath
("./db.sqlite")
conn.Open
Set rs = conn.Execute("SELECT id, name FROM users")

ok = rs.Supports(16)
Response.Write "Supports flag 16: " & CStr(ok)

rs.Close
conn.Close
Set rs = Nothing
Set conn = Nothing
%>
```

481 Recordset.UpdateBatch Method

Applies pending batch updates to the data source.

481.1 Syntax

```
rs.UpdateBatch
```

481.2 Parameters

No parameters.

481.3 Return Value

Empty. The method does not return a value.

481.4 Remarks

- Method names are case-insensitive.
- In the current G3Pix AxonASP ADODB implementation, this method is a compatibility no-op.
- Keep this call for legacy scripts that use batch update flow.

481.5 Code Example

```
<%
Option Explicit
Dim conn, rs

Set conn = Server.CreateObject("ADODB.Connection")
conn.ConnectionString = "Driver={SQLite3};Data Source=" & Server.MapPath
("./db.sqlite")
conn.Open
Set rs = conn.Execute("SELECT id, name FROM users")
```

```
rs.UpdateBatch
Response.Write "UpdateBatch executed"

rs.Close
conn.Close
Set rs = Nothing
Set conn = Nothing
%>
```

482 Recordset.Fields Method

Returns the Fields collection associated with the current recordset.

482.1 Syntax

```
Set fieldsCollection = rs.Fields
```

482.2 Parameters

No parameters.

482.3 Return Value

Object. Returns an ADODB.Fields collection object.

482.4 Remarks

- Method names are case-insensitive.
- Use the collection to inspect schema and read field values.
- Combine with `Fields.Count` and `Fields.Item` for dynamic processing.

482.5 Code Example

```
<%
Option Explicit
Dim conn, rs, fieldsCollection

Set conn = Server.CreateObject("ADODB.Connection")
conn.ConnectionString = "Driver={SQLite3};Data Source=" & Server.MapPath
("./db.sqlite")
conn.Open

Set rs = conn.Execute("SELECT id, name FROM users")
Set fieldsCollection = rs.Fields

Response.Write "Fields: " & CStr(fieldsCollection.Count)

rs.Close
```

```
conn.Close
Set fieldsCollection = Nothing
Set rs = Nothing
Set conn = Nothing
%>
```

483 Recordset.Fields.Append Method

Appends a field definition to the current Recordset Fields collection.

483.1 Syntax

```
rs.Fields.Append name[, type[, definedSize[, attributes]]]
```

483.2 Parameters

Parameter	Type	Required	Description
`name`	String	Yes	Field name to add.
`type`	Integer	No	ADODB type code for the new field.
`definedSize`	Integer	No	Maximum size for variable-length field types.
`attributes`	Integer	No	Field attribute flags.

483.3 Return Value

Empty. The method does not return a value.

483.4 Remarks

- Method names are case-insensitive.
- Use before opening or materializing data when building dynamic structures.
- Field schema changes are provider/runtime dependent.

483.5 Code Example

```
<%
Option Explicit
Dim rs

Set rs = Server.CreateObject("ADODB.Recordset")
rs.Fields.Append "Code", 200, 20
rs.Fields.Append "Description", 200, 255

Response.Write "Fields appended: " & CStr(rs.Fields.Count)
```

```
Set rs = Nothing
%>
```

484 Recordset.Fields.Item Method

Returns a field object from the recordset Fields collection by index or name.

484.1 Syntax

```
Set fld = rs.Fields.Item(indexOrName)
```

484.2 Parameters

Parameter	Type	Required	Description
`indexOrName`	Integer or String	Yes	Zero-based field index or exact field name.

484.3 Return Value

Field. Returns an ADODB.Field object.

484.4 Remarks

- Method names are case-insensitive.
- Invalid name or index raises a runtime error.
- Use index in loops for better performance.

484.5 Code Example

```
<%
Option Explicit
Dim conn, rs, fld

Set conn = Server.CreateObject("ADODB.Connection")
conn.ConnectionString = "Driver={SQLite3};Data Source=" & Server.MapPath
("./db.sqlite")
conn.Open

Set rs = conn.Execute("SELECT id, name FROM users")
If Not rs.EOF Then
    Set fld = rs.Fields.Item("name")
    Response.Write fld.Value
End If

rs.Close
conn.Close
Set fld = Nothing
Set rs = Nothing
```

```
Set conn = Nothing
%>
```

485 Command.Execute Method

Executes the statement configured in `CommandText` using the command connection and parameters.

485.1 Syntax

```
result = cmd.Execute
```

485.2 Parameters

No parameters.

485.3 Return Value

Variant. Returns an ADODB.Recordset for query statements. Returns an Integer with affected rows for action statements. Returns Empty when execution fails.

485.4 Remarks

- Method names are case-insensitive.
- Set `ActiveConnection` and `CommandText` before calling this method.
- Use `CreateParameter` and `Parameters.Append` to pass values safely.

485.5 Code Example

```
<%
Option Explicit
Dim conn, cmd, result

Set conn = Server.CreateObject("ADODB.Connection")
conn.ConnectionString = "Driver={SQLite3};Data Source=" & Server.MapPath
("./db.sqlite")
conn.Open

Set cmd = Server.CreateObject("ADODB.Command")
Set cmd.ActiveConnection = conn
cmd.CommandText = "SELECT id, name FROM users"

Set result = cmd.Execute
If Not result.EOF Then
    Response.Write result.Fields("name").Value
End If

result.Close
conn.Close
```

```
Set result = Nothing
Set cmd = Nothing
Set conn = Nothing
%>
```

486 Command.CreateParameter Method

Creates a Parameter object that can be appended to the command Parameters collection.

486.1 Syntax

```
Set param = cmd.CreateParameter(name, type, direction, size, value)
```

486.2 Parameters

Parameter	Type	Required	Description
`name`	String	No	Parameter name used for readability and provider mapping.
`type`	Integer	No	ADODB data type code.
`direction`	Integer	No	Direction flag: input, output, or input-output.
`size`	Integer	No	Maximum size in bytes or characters.
`value`	Variant	No	Initial parameter value.

486.3 Return Value

Parameter. Returns an ADODB.Parameter object.

486.4 Remarks

- Method names are case-insensitive.
- Call `Parameters.Append` to attach the new parameter to the command.
- CreateParameter does not execute the command.
- Use this method to avoid string concatenation in SQL and improve input safety.

486.5 Code Example

```
<%
Option Explicit
Dim conn, cmd, p, rs

Set conn = Server.CreateObject("ADODB.Connection")
conn.ConnectionString = "Driver={SQLite3};Data Source=" & Server.MapPath
("./db.sqlite")
conn.Open

Set cmd = Server.CreateObject("ADODB.Command")
cmd.ActiveConnection = conn
cmd.CommandText = "SELECT id, name FROM users WHERE id = ?"

Set p = cmd.CreateParameter("id", 3, 1, 4, 1)
cmd.Parameters.Append p

Set rs = cmd.Execute
If Not rs.EOF Then
    Response.Write rs.Fields("name").Value
End If

rs.Close
conn.Close
Set rs = Nothing
Set p = Nothing
Set cmd = Nothing
Set conn = Nothing
%>
```

487 Command.Cancel Method

Cancels the current command execution.

487.1 Syntax

```
cmd.Cancel
```

487.2 Parameters

No parameters.

487.3 Return Value

Empty. The method does not return a value.

487.4 Remarks

- Method names are case-insensitive.

- In the current G3Pix AxonASP ADODB implementation, this method is a compatibility no-op.
- Calling Cancel does not interrupt already completed operations.
- Use this method only for compatibility with scripts that expect the ADODB command surface.

487.5 Code Example

```
<%
Option Explicit
Dim conn, cmd

Set conn = Server.CreateObject("ADODB.Connection")
conn.ConnectionString = "Driver={SQLite3};Data Source=" & Server.MapPath
("./db.sqlite")
conn.Open

Set cmd = Server.CreateObject("ADODB.Command")
cmd.ActiveConnection = conn
cmd.CommandText = "SELECT id, name FROM users"

' Compatibility no-op in current runtime.
cmd.Cancel

Response.Write "Command.Cancel executed"

conn.Close
Set cmd = Nothing
Set conn = Nothing
%>
```

488 Command.Parameters.Refresh Method

Refreshes the command parameter metadata from the provider.

488.1 Syntax

```
cmd.Parameters.Refresh
```

488.2 Parameters

No parameters.

488.3 Return Value

Empty. The method does not return a value.

488.4 Remarks

- Method names are case-insensitive.
- In the current G3Pix AxonASP ADODB implementation, this method is a compatibility no-op.
- Use explicit `CreateParameter` and `Parameters.Append` when defining parameters.
- Call this method only for compatibility with existing scripts.

488.5 Code Example

```
<%
Option Explicit
Dim conn, cmd

Set conn = Server.CreateObject("ADODB.Connection")
conn.ConnectionString = "Driver={SQLite3};Data Source=" & Server.MapPath
("./db.sqlite")
conn.Open

Set cmd = Server.CreateObject("ADODB.Command")
cmd.ActiveConnection = conn
cmd.CommandText = "SELECT id FROM users WHERE id = ?"

' Compatibility no-op in current runtime.
cmd.Parameters.Refresh

Response.Write "Parameters.Refresh executed"

conn.Close
Set cmd = Nothing
Set conn = Nothing
%>
```

489 Errors.Count Method

Returns the number of error objects in the Errors collection.

489.1 Syntax

```
count = conn.Errors.Count
```

489.2 Parameters

No parameters.

489.3 Return Value

Integer. Returns the number of errors currently stored in the connection Errors collection.

489.4 Remarks

- Method names are case-insensitive.
- The Errors collection is associated with a Connection object.
- The count increases when an operation fails and adds an ADODB.Error item.
- Call `Errors.Clear` to reset the collection.

489.5 Code Example

```
<%
Option Explicit
Dim conn

Set conn = Server.CreateObject("ADODB.Connection")
conn.ConnectionString = "Driver={SQLite3};Data Source=" & Server.MapPath
("./db.sqlite")
conn.Open

' Force an error to populate Errors collection.
On Error Resume Next
conn.Execute "SELECT * FROM table_that_does_not_exist"
On Error GoTo 0

Response.Write "Errors count: " & CStr(conn.Errors.Count)

conn.Close
Set conn = Nothing
%>
```

490 Errors.Item Method

Returns an Error object from the Errors collection by index.

490.1 Syntax

```
Set errObj = conn.Errors.Item(index)
```

490.2 Parameters

Parameter	Type	Required	Description
`index`	Integer	Yes	Zero-based index of the error item in the collection.

490.3 Return Value

Error or Empty. Returns an ADODB.Error object for a valid index, or Empty when the index is out of range.

490.4 Remarks

- Method names are case-insensitive.
- Use `Errors.Count` to validate the index before calling Item.
- Error objects expose Number, Description, Source, and SQLState.
- The Errors collection belongs to the Connection object.

490.5 Code Example

```
<%
Option Explicit
Dim conn, errObj

Set conn = Server.CreateObject("ADODB.Connection")
conn.ConnectionString = "Driver={SQLite3};Data Source=" & Server.MapPath
("./db.sqlite")
conn.Open

On Error Resume Next
conn.Execute "SELECT * FROM table_that_does_not_exist"
On Error GoTo 0

If conn.Errors.Count > 0 Then
    Set errObj = conn.Errors.Item(0)
    Response.Write "Error " & CStr(errObj.Number) & ": " & errObj.Descrip
tion
End If

conn.Close
Set errObj = Nothing
Set conn = Nothing
%>
```

491 Errors.Clear Method

Clears all error entries from the connection Errors collection.

491.1 Syntax

```
conn.Errors.Clear
```

491.2 Parameters

No parameters.

491.3 Return Value

Empty. The method does not return a value.

491.4 Remarks

- Method names are case-insensitive.
- Use Clear after handling errors to reset the collection state.
- Clear does not fix the source problem; it only removes stored error entries.
- The Errors collection belongs to the current Connection object.

491.5 Code Example

```
<%
Option Explicit
Dim conn

Set conn = Server.CreateObject("ADODB.Connection")
conn.ConnectionString = "Driver={SQLite3};Data Source=" & Server.MapPath
("./db.sqlite")
conn.Open

On Error Resume Next
conn.Execute "SELECT * FROM table_that_does_not_exist"
On Error GoTo 0

Response.Write "Before clear: " & CStr(conn.Errors.Count) & "<br>"
conn.Errors.Clear
Response.Write "After clear: " & CStr(conn.Errors.Count)

conn.Close
Set conn = Nothing
%>
```

492 Fields.Append Method

Appends a new field definition to a Fields collection.

492.1 Syntax

```
fields.Append name[, type[, definedSize[, attributes]]]
```

492.2 Parameters

Parameter	Type	Required	Description
name	String	Yes	Field name to append.
type	Integer	No	ADODB data type

Parameter	Type	Required	Description
			code.
`definedSize`	Integer	No	Declared size for variable-length fields.
`attributes`	Integer	No	Field attribute bitmask.

492.3 Return Value

Empty. The method does not return a value.

492.4 Remarks

- Method names are case-insensitive.
- Append is used for dynamic schema creation scenarios.
- Provider support for runtime schema mutation can vary.

492.5 Code Example

```
<%
Option Explicit
Dim rs

Set rs = Server.CreateObject("ADODB.Recordset")
rs.Fields.Append "TempName", 200, 100

Response.Write "Field appended"

Set rs = Nothing
%>
```

493 Fields.Item Method

Returns a Field object by index or column name.

493.1 Syntax

```
Set fieldObj = rs.Fields.Item(indexOrName)
```

493.2 Parameters

Parameter	Type	Required	Description
`indexOrName`	Integer or String	Yes	Zero-based field index or exact field name.

493.3 Return Value

Field. Returns an ADODB.Field object for the requested column.

493.4 Remarks

- Method names are case-insensitive.
- Use index for faster access in tight loops.
- Use name for readable code when column order can change.
- Invalid index or name raises a runtime error.

493.5 Code Example

```
<%
Option Explicit
Dim conn, rs, fldByIndex, fldByName

Set conn = Server.CreateObject("ADODB.Connection")
conn.ConnectionString = "Driver={SQLite3};Data Source=" & Server.MapPath
("./db.sqlite")
conn.Open

Set rs = conn.Execute("SELECT id, name FROM users")
If Not rs.EOF Then
    Set fldByIndex = rs.Fields.Item(0)
    Set fldByName = rs.Fields.Item("name")
    Response.Write "Field[0]: " & fldByIndex.Value & "<br>"
    Response.Write "Field['name']: " & fldByName.Value
End If

rs.Close
conn.Close
Set fldByName = Nothing
Set fldByIndex = Nothing
Set rs = Nothing
Set conn = Nothing
%>
```

494 Fields.Count Method

Returns the number of columns in the Fields collection.

494.1 Syntax

```
count = rs.Fields.Count
```

494.2 Parameters

No parameters.

494.3 Return Value

Integer. Returns the number of Field objects in the current Recordset schema.

494.4 Remarks

- Method names are case-insensitive.
- Count reflects the selected columns in the current query.
- The value is available after the Recordset is opened.
- Use Count to iterate safely over the fields collection.

494.5 Code Example

```
<%
Option Explicit
Dim conn, rs, i

Set conn = Server.CreateObject("ADODB.Connection")
conn.ConnectionString = "Driver={SQLite3};Data Source=" & Server.MapPath
("./db.sqlite")
conn.Open

Set rs = conn.Execute("SELECT id, name, email FROM users")
Response.Write "Field count: " & CStr(rs.Fields.Count) & "<br>"

For i = 0 To rs.Fields.Count - 1
    Response.Write "Column " & CStr(i) & ": " & rs.Fields.Item(i).Name &
"<br>"
Next

rs.Close
conn.Close
Set rs = Nothing
Set conn = Nothing
%>
```

495 Field.AppendChunk Method

Appends binary or text chunk data to the current field value.

495.1 Syntax

```
fieldObj.AppendChunk data
```

495.2 Parameters

Parameter	Type	Required	Description
`data`	String or Byte Array	Yes	Chunk data appended to the

Parameter	Type	Required	Description
			target field.

495.3 Return Value

Empty. The method does not return a value.

495.4 Remarks

- Method names are case-insensitive.
- AppendChunk is commonly used for long text and binary fields.
- Multiple AppendChunk calls append data sequentially.
- Call `Recordset.Update` to persist appended data.

495.5 Code Example

```
<%
Option Explicit
Dim conn, rs

Set conn = Server.CreateObject("ADODB.Connection")
conn.ConnectionString = "Driver={SQLite3};Data Source=" & Server.MapPath
("./db.sqlite")
conn.Open

Set rs = conn.Execute("SELECT * FROM users WHERE id = 1")
If Not rs.EOF Then
    rs.Fields("notes").AppendChunk " First chunk."
    rs.Fields("notes").AppendChunk " Second chunk."
    rs.Update
End If

rs.Close
conn.Close
Set rs = Nothing
Set conn = Nothing
%>
```

496 Field.GetChunk Method

Reads a chunk of data from the current field value.

496.1 Syntax

```
chunk = fieldObj.GetChunk(length)
```

496.2 Parameters

Parameter	Type	Required	Description

Parameter	Type	Required	Description
length	Integer	No	Number of bytes or characters to read. If omitted, reads remaining data.

496.3 Return Value

String. Returns the chunk data read from the field.

496.4 Remarks

- Method names are case-insensitive.
- Use GetChunk to read long text or binary content in segments.
- Repeated calls return subsequent portions of the field data.
- Reset reading position by reloading the row.

496.5 Code Example

```
<%
Option Explicit
Dim conn, rs, part

Set conn = Server.CreateObject("ADODB.Connection")
conn.ConnectionString = "Driver={SQLite3};Data Source=" & Server.MapPath
("./db.sqlite")
conn.Open

Set rs = conn.Execute("SELECT notes FROM users WHERE id = 1")
If Not rs.EOF Then
    part = rs.Fields("notes").GetChunk(20)
    Response.Write "Chunk: " & part
End If

rs.Close
conn.Close
Set rs = Nothing
Set conn = Nothing
%>
```

497 Parameters.Append Method

Adds a Parameter object to the command Parameters collection.

497.1 Syntax

```
cmd.Parameters.Append parameterObj
```

497.2 Parameters

Parameter	Type	Required	Description
`parameterObj`	Parameter	Yes	Parameter object created by `CreateParameter`.

497.3 Return Value

Empty. The method does not return a value.

497.4 Remarks

- Method names are case-insensitive.
- Append adds the parameter to the end of the collection.
- Parameter order must match placeholder order for positional providers.
- Append does not execute the command.

497.5 Code Example

```
<%
Option Explicit
Dim conn, cmd, p

Set conn = Server.CreateObject("ADODB.Connection")
conn.ConnectionString = "Driver={SQLite3};Data Source=" & Server.MapPath
("./db.sqlite")
conn.Open

Set cmd = Server.CreateObject("ADODB.Command")
cmd.ActiveConnection = conn
cmd.CommandText = "SELECT id, name FROM users WHERE id = ?"

Set p = cmd.CreateParameter("id", 3, 1, 4, 1)
cmd.Parameters.Append p

Response.Write "Parameters count: " & CStr(cmd.Parameters.Count)

conn.Close
Set p = Nothing
Set cmd = Nothing
Set conn = Nothing
%>
```

498 Parameters.Count Method

Returns the number of Parameter objects in the command Parameters collection.

498.1 Syntax

```
count = cmd.Parameters.Count
```

498.2 Parameters

No parameters.

498.3 Return Value

Integer. Returns the number of parameters currently appended to the command.

498.4 Remarks

- Method names are case-insensitive.
- Use Count to validate parameter setup before Execute.
- Count reflects only parameters explicitly appended in the current command object.
- Parameter order matters for positional placeholders.

498.5 Code Example

```
<%
Option Explicit
Dim conn, cmd, p

Set conn = Server.CreateObject("ADODB.Connection")
conn.ConnectionString = "Driver={SQLite3};Data Source=" & Server.MapPath
("./db.sqlite")
conn.Open

Set cmd = Server.CreateObject("ADODB.Command")
Set cmd.ActiveConnection = conn
cmd.CommandText = "SELECT id FROM users WHERE id = ?"

Set p = cmd.CreateParameter("id", 3, 1, 4, 1)
cmd.Parameters.Append p

Response.Write "Parameters count: " & CStr(cmd.Parameters.Count)

conn.Close
Set p = Nothing
Set cmd = Nothing
Set conn = Nothing
%>
```

499 Connection.ConnectionString Property

Gets or sets the full database connection string.

499.1 Syntax

```
text = conn.ConnectionString
conn.ConnectionString = "Driver={SQLite3};Data Source=C:\\data\\app.db"
```

499.2 Return Value

String. Returns the current connection string text.

499.3 Remarks

- Property names are case-insensitive.
- Set ConnectionString before calling `Connection.Open`.
- Include provider and data source in the value.
- Invalid connection strings raise provider errors when opening the connection.

499.4 Code Example

```
<%
Option Explicit
Dim conn

Set conn = Server.CreateObject("ADODB.Connection")
conn.ConnectionString = "Driver={SQLite3};Data Source=" & Server.MapPath
("./db.sqlite")
Response.Write "Configured: " & conn.ConnectionString

conn.Open
Response.Write "<br>State: " & CStr(conn.State)

conn.Close
Set conn = Nothing
%>
```

500 Connection.State Property

Returns the current open/closed state of the connection.

500.1 Syntax

```
state = conn.State
```

500.2 Return Value

Integer. Returns `0` when closed and `1` when open.

500.3 Remarks

- Property names are case-insensitive.
- This property is read-only.

- Check this value before operations that require an open connection.

500.4 Code Example

```
<%
Option Explicit
Dim conn

Set conn = Server.CreateObject("ADODB.Connection")
Response.Write "Before open: " & CStr(conn.State) & "<br>"

conn.ConnectionString = "Driver={SQLite3};Data Source=" & Server.MapPath
("./db.sqlite")
conn.Open
Response.Write "After open: " & CStr(conn.State) & "<br>"

conn.Close
Response.Write "After close: " & CStr(conn.State)

Set conn = Nothing
%>
```

501 Connection.Mode Property

Gets or sets the access mode for the connection.

501.1 Syntax

```
mode = conn.Mode
conn.Mode = 3
```

501.2 Return Value

Integer. Returns the current mode flag.

501.3 Remarks

- Property names are case-insensitive.
- Typical flags include read-only and read-write combinations.
- Set Mode before calling Open.
- Unsupported mode flags can be ignored or rejected by provider.

501.4 Code Example

```
<%
Option Explicit
Dim conn

Set conn = Server.CreateObject("ADODB.Connection")
conn.Mode = 3
```

```
Response.Write "Mode: " & CStr(conn.Mode)

Set conn = Nothing
%>
```

502 Connection.Provider Property

Gets or sets the provider name used by the connection.

502.1 Syntax

```
name = conn.Provider
conn.Provider = "MSDASQL"
```

502.2 Return Value

String. Returns the provider identifier.

502.3 Remarks

- Property names are case-insensitive.
- Provider can be set directly or inferred from ConnectionString.
- When both are set, provider resolution depends on the active driver/provider rules.
- Use a provider supported by your host environment.

502.4 Code Example

```
<%
Option Explicit
Dim conn

Set conn = Server.CreateObject("ADODB.Connection")
conn.Provider = "MSDASQL"
conn.ConnectionString = "Driver={SQLite3};Data Source=" & Server.MapPath
("./db.sqlite")

Response.Write "Provider: " & conn.Provider

Set conn = Nothing
%>
```

503 Connection.Version Property

Returns the ADODB compatibility version string exposed by the runtime.

503.1 Syntax

```
text = conn.Version
```

503.2 Return Value

String. Returns the version identifier for the ADODB compatibility surface.

503.3 Remarks

- Property names are case-insensitive.
- This property is read-only.
- Use this value for diagnostics and compatibility reporting.
- The value does not represent the underlying database engine version.

503.4 Code Example

```
<%
Option Explicit
Dim conn

Set conn = Server.CreateObject("ADODB.Connection")
Response.Write "ADODB compatibility version: " & conn.Version

Set conn = Nothing
%>
```

504 Connection.CommandTimeout Property

Gets or sets the default timeout for commands executed through the connection.

504.1 Syntax

```
seconds = conn.CommandTimeout
conn.CommandTimeout = 30
```

504.2 Return Value

Integer. Returns the command timeout in seconds.

504.3 Remarks

- Property names are case-insensitive.
- Applies to commands executed via `Connection.Execute`.
- Set this property before running long or uncertain queries.
- Individual `Command.CommandTimeout` values can override this default.

504.4 Code Example

```
<%
Option Explicit
Dim conn
```

```
Set conn = Server.CreateObject("ADODB.Connection")
conn.ConnectionString = "Driver={SQLite3};Data Source=" & Server.MapPath
("./db.sqlite")
conn.Open
conn.CommandTimeout = 20

Response.Write "Command timeout: " & CStr(conn.CommandTimeout) & " second
s"

conn.Close
Set conn = Nothing
%>
```

505 Connection.ConnectionTimeout Property

Gets or sets the timeout for establishing the connection, in seconds.

505.1 Syntax

```
seconds = conn.ConnectionTimeout
conn.ConnectionTimeout = 15
```

505.2 Return Value

Integer. Returns the configured connection timeout in seconds.

505.3 Remarks

- Property names are case-insensitive.
- Set this property before calling `Connection.Open`.
- Lower values fail faster on unavailable endpoints.
- Very low values can cause failures on slow networks or startup-heavy providers.

505.4 Code Example

```
<%
Option Explicit
Dim conn

Set conn = Server.CreateObject("ADODB.Connection")
conn.ConnectionString = "Driver={SQLite3};Data Source=" & Server.MapPath
("./db.sqlite")
conn.ConnectionTimeout = 10

Response.Write "Connection timeout: " & CStr(conn.ConnectionTimeout) & "
seconds"

conn.Open
```

```
conn.Close
Set conn = Nothing
%>
```

506 Connection.CursorLocation Property

Gets or sets the cursor location mode used by the connection.

506.1 Syntax

```
mode = conn.CursorLocation
conn.CursorLocation = 3
```

506.2 Return Value

Integer. Returns the configured cursor location flag.

506.3 Remarks

- Property names are case-insensitive.
- Common values are 1 (server cursor) and 3 (client cursor).
- Set this property before opening recordsets that inherit connection defaults.
- Individual recordsets can override cursor location.

506.4 Code Example

```
<%
Option Explicit
Dim conn

Set conn = Server.CreateObject("ADODB.Connection")
conn.CursorLocation = 3
Response.Write "CursorLocation: " & CStr(conn.CursorLocation)

Set conn = Nothing
%>
```

507 Connection.DefaultDatabase Property

Gets or sets the default database or catalog used by the connection.

507.1 Syntax

```
name = conn.DefaultDatabase
conn.DefaultDatabase = "main"
```

507.2 Return Value

String. Returns the default database name.

507.3 Remarks

- Property names are case-insensitive.
- Use this property to control unqualified table resolution.
- Effective behavior depends on the underlying provider.
- Set before running queries that rely on default catalog context.

507.4 Code Example

```
<%
Option Explicit
Dim conn

Set conn = Server.CreateObject("ADODB.Connection")
conn.DefaultDatabase = "main"
Response.Write "DefaultDatabase: " & conn.DefaultDatabase

Set conn = Nothing
%>
```

508 Connection.IsolationLevel Property

Gets or sets the transaction isolation level used for new transactions.

508.1 Syntax

```
level = conn.IsolationLevel
conn.IsolationLevel = 4096
```

508.2 Return Value

Integer. Returns the configured isolation level flag.

508.3 Remarks

- Property names are case-insensitive.
- Isolation level affects locking and visibility rules during transactions.
- Set before calling BeginTrans.
- Provider support for specific isolation flags can vary.

508.4 Code Example

```
<%
Option Explicit
Dim conn

Set conn = Server.CreateObject("ADODB.Connection")
conn.IsolationLevel = 4096
Response.Write "IsolationLevel: " & CStr(conn.IsolationLevel)
```

```
Set conn = Nothing
%>
```

509 Connection.Errors Property

Returns the Errors collection associated with the connection.

509.1 Syntax

```
Set errs = conn.Errors
```

509.2 Return Value

Object. Returns an ADODB.Errors collection object.

509.3 Remarks

- Property names are case-insensitive.
- This property is read-only.
- Use `Errors.Count`, `Errors.Item`, and `Errors.Clear` to inspect and manage error entries.
- The collection is updated after failed connection and command operations.

509.4 Code Example

```
<%
Option Explicit
Dim conn, errs

Set conn = Server.CreateObject("ADODB.Connection")
conn.ConnectionString = "Driver={SQLite3};Data Source=" & Server.MapPath
("./db.sqlite")
conn.Open

On Error Resume Next
conn.Execute "SELECT * FROM table_that_does_not_exist"
On Error GoTo 0

Set errs = conn.Errors
Response.Write "Errors count: " & CStr(errs.Count)

conn.Close
Set errs = Nothing
Set conn = Nothing
%>
```

510 Recordset.EOF Property

Indicates whether the cursor is positioned after the last row.

510.1 Syntax

```
eofFlag = rs.EOF
```

510.2 Return Value

Boolean. Returns True when cursor is beyond the final row; otherwise False.

510.3 Remarks

- Property names are case-insensitive.
- This property is read-only.
- Combine with BOF to detect empty result sets.

510.4 Code Example

```
<%
Option Explicit
Dim conn, rs

Set conn = Server.CreateObject("ADODB.Connection")
conn.ConnectionString = "Driver={SQLite3};Data Source=" & Server.MapPath
("./db.sqlite")
conn.Open
Set rs = conn.Execute("SELECT id, name FROM users")

Do While Not rs.EOF
    Response.Write rs.Fields("name").Value & "<br>"
    rs.MoveNext
Loop

rs.Close
conn.Close
Set rs = Nothing
Set conn = Nothing
%>
```

511 Recordset.BOF Property

Indicates whether the cursor is before the first row.

511.1 Syntax

```
flag = rs.BOF
```

511.2 Return Value

Boolean. Returns True when cursor is before the first row; otherwise False.

511.3 Remarks

- Property names are case-insensitive.
- This property is read-only.
- Use together with EOF to detect empty rowsets.

511.4 Code Example

```
<%
Option Explicit
Dim conn, rs

Set conn = Server.CreateObject("ADODB.Connection")
conn.ConnectionString = "Driver={SQLite3};Data Source=" & Server.MapPath
("./db.sqlite")
conn.Open
Set rs = conn.Execute("SELECT id, name FROM users")

Response.Write "BOF: " & CStr(rs.BOF)

rs.Close
conn.Close
Set rs = Nothing
Set conn = Nothing
%>
```

512 Recordset.State Property

Returns the open/closed state of the recordset.

512.1 Syntax

```
value = rs.State
```

512.2 Return Value

Integer. Returns `0` when closed and `1` when open.

512.3 Remarks

- Property names are case-insensitive.
- This property is read-only.
- Check before reading fields or moving the cursor.

512.4 Code Example

```
<%
Option Explicit
Dim conn, rs

Set conn = Server.CreateObject("ADODB.Connection")
conn.ConnectionString = "Driver={SQLite3};Data Source=" & Server.MapPath
("./db.sqlite")
conn.Open

Set rs = conn.Execute("SELECT id, name FROM users")
Response.Write "Open state: " & CStr(rs.State) & "<br>"

rs.Close
Response.Write "Closed state: " & CStr(rs.State)

conn.Close
Set rs = Nothing
Set conn = Nothing
%>
```

513 Recordset.RecordCount Property

Returns the number of rows currently represented by the recordset.

513.1 Syntax

```
count = rs.RecordCount
```

513.2 Return Value

Integer. Returns row count, or provider-specific fallback when exact count is unavailable.

513.3 Remarks

- Property names are case-insensitive.
- This property is read-only.
- Some cursor/provider combinations may require movement to finalize count.

513.4 Code Example

```
<%
Option Explicit
Dim conn, rs

Set conn = Server.CreateObject("ADODB.Connection")
conn.ConnectionString = "Driver={SQLite3};Data Source=" & Server.MapPath
("./db.sqlite")
```

```
conn.Open
Set rs = conn.Execute("SELECT id, name FROM users")

Response.Write CStr(rs.RecordCount)

rs.Close
conn.Close
Set rs = Nothing
Set conn = Nothing
%>
```

514 Recordset.Fields Property

Returns the Fields collection for the current recordset schema.

514.1 Syntax

```
Set fieldsCollection = rs.Fields
```

514.2 Return Value

Object. Returns an ADODB.Fields collection.

514.3 Remarks

- Property names are case-insensitive.
- This property is read-only.
- Use `Fields.Count` and `Fields.Item` to inspect columns.

514.4 Code Example

```
<%
Option Explicit
Dim conn, rs, fieldsCollection

Set conn = Server.CreateObject("ADODB.Connection")
conn.ConnectionString = "Driver={SQLite3};Data Source=" & Server.MapPath
("./db.sqlite")
conn.Open
Set rs = conn.Execute("SELECT id, name FROM users")

Set fieldsCollection = rs.Fields
Response.Write CStr(fieldsCollection.Count)

rs.Close
conn.Close
Set fieldsCollection = Nothing
Set rs = Nothing
Set conn = Nothing
%>
```

515 Recordset.AbsolutePage Property

Gets or sets the current page number in paged recordset navigation.

515.1 Syntax

```
page = rs.AbsolutePage
rs.AbsolutePage = newPage
```

515.2 Return Value

Integer. Returns the current page index.

515.3 Remarks

- Property names are case-insensitive.
- Page behavior depends on `PageSize` and provider support.
- Set `PageSize` before relying on page-based navigation.

515.4 Code Example

```
<%
Option Explicit
Dim conn, rs

Set conn = Server.CreateObject("ADODB.Connection")
conn.ConnectionString = "Driver={SQLite3};Data Source=" & Server.MapPath
("./db.sqlite")
conn.Open
Set rs = conn.Execute("SELECT id, name FROM users")

rs.PageSize = 10
Response.Write "Page: " & CStr(rs.AbsolutePage)

rs.Close
conn.Close
Set rs = Nothing
Set conn = Nothing
%>
```

516 Recordset.PageCount Property

Returns the total number of pages based on row count and page size.

516.1 Syntax

```
value = rs.PageCount
```

516.2 Return Value

Integer. Returns computed total pages.

516.3 Remarks

- Property names are case-insensitive.
- This property is read-only.
- Depends on `PageSize` and row materialization state.

516.4 Code Example

```
<%
Option Explicit
Dim conn, rs

Set conn = Server.CreateObject("ADODB.Connection")
conn.ConnectionString = "Driver={SQLite3};Data Source=" & Server.MapPath
("./db.sqlite")
conn.Open
Set rs = conn.Execute("SELECT id, name FROM users")

rs.PageSize = 10
Response.Write CStr(rs.PageCount)

rs.Close
conn.Close
Set rs = Nothing
Set conn = Nothing
%>
```

517 Recordset.PageSize Property

Gets or sets the number of rows per logical page.

517.1 Syntax

```
value = rs.PageSize
rs.PageSize = newValue
```

517.2 Return Value

Integer. Returns configured page size.

517.3 Remarks

- Property names are case-insensitive.
- Set this value before page-based navigation.
- A value less than 1 is invalid for page calculations.

517.4 Code Example

```
<%
Option Explicit
Dim rs

Set rs = Server.CreateObject("ADODB.Recordset")
rs.PageSize = 25
Response.Write CStr(rs.PageSize)

Set rs = Nothing
%>
```

518 Recordset.AbsolutePosition Property

Gets or sets the current row position in the recordset.

518.1 Syntax

```
pos = rs.AbsolutePosition
rs.AbsolutePosition = newPos
```

518.2 Return Value

Integer. Returns the current one-based row position.

518.3 Remarks

- Property names are case-insensitive.
- Position support depends on cursor type/provider behavior.
- Assigning out-of-range values can move cursor to BOF/EOF boundaries.

518.4 Code Example

```
<%
Option Explicit
Dim conn, rs

Set conn = Server.CreateObject("ADODB.Connection")
conn.ConnectionString = "Driver={SQLite3};Data Source=" & Server.MapPath
("./db.sqlite")
conn.Open
Set rs = conn.Execute("SELECT id, name FROM users")

rs.MoveFirst
Response.Write "Position: " & CStr(rs.AbsolutePosition)

rs.Close
conn.Close
Set rs = Nothing
```

```
Set conn = Nothing
%>
```

519 Recordset.Bookmark Property

Gets or sets a bookmark that identifies the current row position.

519.1 Syntax

```
bm = rs.Bookmark
rs.Bookmark = bm
```

519.2 Return Value

Variant. Returns the bookmark token for the current row.

519.3 Remarks

- Property names are case-insensitive.
- Bookmark support depends on cursor/provider capabilities.
- Save bookmark values to return to a previously visited row.

519.4 Code Example

```
<%
Option Explicit
Dim conn, rs, bm

Set conn = Server.CreateObject("ADODB.Connection")
conn.ConnectionString = "Driver={SQLite3};Data Source=" & Server.MapPath
("./db.sqlite")
conn.Open
Set rs = conn.Execute("SELECT id, name FROM users")

If Not rs.EOF Then
    bm = rs.Bookmark
    rs.MoveNext
    rs.Bookmark = bm
    Response.Write rs.Fields("name").Value
End If

rs.Close
conn.Close
Set rs = Nothing
Set conn = Nothing
%>
```

520 Recordset.ActiveCommand Property

Returns or sets the command object associated with the recordset.

520.1 Syntax

```
Set cmdRef = rs.ActiveCommand
Set rs.ActiveCommand = cmd
```

520.2 Return Value

Object. Returns an ADODB.Command reference when available.

520.3 Remarks

- Property names are case-insensitive.
- Use Set for assignment and retrieval.
- This association is useful for introspection of command-driven recordsets.

520.4 Code Example

```
<%
Option Explicit
Dim conn, cmd, rs, cmdRef

Set conn = Server.CreateObject("ADODB.Connection")
conn.ConnectionString = "Driver={SQLite3};Data Source=" & Server.MapPath
("./db.sqlite")
conn.Open

Set cmd = Server.CreateObject("ADODB.Command")
Set cmd.ActiveConnection = conn
cmd.CommandText = "SELECT id, name FROM users"
Set rs = cmd.Execute
Set cmdRef = rs.ActiveCommand

If IsObject(cmdRef) Then Response.Write "ActiveCommand available"

rs.Close
conn.Close
Set cmdRef = Nothing
Set rs = Nothing
Set cmd = Nothing
Set conn = Nothing
%>
```

521 Recordset.CacheSize Property

Gets or sets the number of rows cached locally for cursor operations.

521.1 Syntax

```
value = rs.CacheSize
rs.CacheSize = newValue
```

521.2 Return Value

Integer. Returns configured cache size.

521.3 Remarks

- Property names are case-insensitive.
- Set before heavy navigation workloads.
- Effective behavior can depend on provider cursor implementation.

521.4 Code Example

```
<%
Option Explicit
Dim conn, rs

Set conn = Server.CreateObject("ADODB.Connection")
conn.ConnectionString = "Driver={SQLite3};Data Source=" & Server.MapPath
("./db.sqlite")
conn.Open
Set rs = conn.Execute("SELECT id, name FROM users")

rs.CacheSize = 25
Response.Write CStr(rs.CacheSize)

rs.Close
conn.Close
Set rs = Nothing
Set conn = Nothing
%>
```

522 Recordset.DataMember Property

Gets or sets the named data member associated with the recordset.

522.1 Syntax

```
value = rs.DataMember
rs.DataMember = newValue
```

522.2 Return Value

String. Returns current data member name.

522.3 Remarks

- Property names are case-insensitive.
- Used mainly in bound-data scenarios.
- Often empty for direct SQL recordsets.

522.4 Code Example

```
<%
Option Explicit
Dim rs

Set rs = Server.CreateObject("ADODB.Recordset")
rs.DataMember = "Users"
Response.Write rs.DataMember

Set rs = Nothing
%>
```

523 Recordset.EditMode Property

Returns the current edit state for the active row.

523.1 Syntax

```
value = rs.EditMode
```

523.2 Return Value

Integer. Returns edit mode flag (for example, not editing, editing, or add-new state).

523.3 Remarks

- Property names are case-insensitive.
- This property is read-only.
- Use to detect pending edits before `Update` or `CancelUpdate`.

523.4 Code Example

```
<%
Option Explicit
Dim conn, rs

Set conn = Server.CreateObject("ADODB.Connection")
conn.ConnectionString = "Driver={SQLite3};Data Source=" & Server.MapPath
("./db.sqlite")
conn.Open
Set rs = conn.Execute("SELECT * FROM users WHERE id = 1")

If Not rs.EOF Then
    Response.Write "Before: " & CStr(rs.EditMode) & "<br>"
    rs.Fields("name").Value = "Temp"
    Response.Write "After: " & CStr(rs.EditMode)
End If

rs.Close
```

```
conn.Close
Set rs = Nothing
Set conn = Nothing
%>
```

524 Recordset.Filter Property

Gets or sets a filter expression for visible rows in the recordset.

524.1 Syntax

```
value = rs.Filter
rs.Filter = newFilter
```

524.2 Return Value

String or Integer. Returns the current filter setting.

524.3 Remarks

- Property names are case-insensitive.
- Filter syntax support depends on provider/runtime capabilities.
- Use simple predicates for maximum compatibility.

524.4 Code Example

```
<%
Option Explicit
Dim conn, rs

Set conn = Server.CreateObject("ADODB.Connection")
conn.ConnectionString = "Driver={SQLite3};Data Source=" & Server.MapPath
("./db.sqlite")
conn.Open
Set rs = conn.Execute("SELECT id, name FROM users")

rs.Filter = "id = 1"
If Not rs.EOF Then Response.Write rs.Fields("name").Value

rs.Close
conn.Close
Set rs = Nothing
Set conn = Nothing
%>
```

525 Recordset.Index Property

Gets or sets the active index name used by seek operations.

525.1 Syntax

```
value = rs.Index
rs.Index = newIndex
```

525.2 Return Value

String. Returns the current index name.

525.3 Remarks

- Property names are case-insensitive.
- Set this before calling `Recordset.Seek` when index-based navigation is needed.
- Provider support for named indexes can vary.

525.4 Code Example

```
<%
Option Explicit
Dim conn, rs

Set conn = Server.CreateObject("ADODB.Connection")
conn.ConnectionString = "Driver={SQLite3};Data Source=" & Server.MapPath
("./db.sqlite")
conn.Open
Set rs = conn.Execute("SELECT id, name FROM users")

rs.Index = "id"
Response.Write rs.Index

rs.Close
conn.Close
Set rs = Nothing
Set conn = Nothing
%>
```

526 Recordset.MarshalOptions Property

Gets or sets marshaling behavior flags for recordset transport.

526.1 Syntax

```
value = rs.MarshalOptions
rs.MarshalOptions = newValue
```

526.2 Return Value

Integer. Returns current marshal options flag.

526.3 Remarks

- Property names are case-insensitive.
- Marshaling behavior can be provider/runtime specific.
- Set before serialization or transport scenarios.

526.4 Code Example

```
<%
Option Explicit
Dim rs

Set rs = Server.CreateObject("ADODB.Recordset")
rs.MarshalOptions = 0
Response.Write CStr(rs.MarshalOptions)

Set rs = Nothing
%>
```

527 Recordset.MaxRecords Property

Gets or sets the maximum number of rows the recordset should return.

527.1 Syntax

```
value = rs.MaxRecords
rs.MaxRecords = newValue
```

527.2 Return Value

Integer. Returns maximum rows limit.

527.3 Remarks

- Property names are case-insensitive.
- Set this before opening the recordset.
- `0` usually means no explicit limit.

527.4 Code Example

```
<%
Option Explicit
Dim rs

Set rs = Server.CreateObject("ADODB.Recordset")
rs.MaxRecords = 100
Response.Write CStr(rs.MaxRecords)

Set rs = Nothing
%>
```

528 Recordset.Sort Property

Gets or sets the sort expression applied to recordset rows.

528.1 Syntax

```
value = rs.Sort
rs.Sort = "columnName ASC"
```

528.2 Return Value

String. Returns the current sort expression.

528.3 Remarks

- Property names are case-insensitive.
- Sort expression format is provider/runtime dependent.
- Use valid field names present in the recordset schema.

528.4 Code Example

```
<%
Option Explicit
Dim conn, rs

Set conn = Server.CreateObject("ADODB.Connection")
conn.ConnectionString = "Driver={SQLite3};Data Source=" & Server.MapPath
("./db.sqlite")
conn.Open
Set rs = conn.Execute("SELECT id, name FROM users")

rs.Sort = "name ASC"
Response.Write rs.Sort

rs.Close
conn.Close
Set rs = Nothing
Set conn = Nothing
%>
```

529 Recordset.Source Property

Gets or sets the source used to populate the recordset.

529.1 Syntax

```
value = rs.Source
rs.Source = "SELECT id, name FROM users"
```

529.2 Return Value

Variant. Returns the source expression, typically SQL text or command reference.

529.3 Remarks

- Property names are case-insensitive.
- Set Source before opening the recordset.
- Source can be SQL text, table name, or a command object depending on runtime path.

529.4 Code Example

```
<%
Option Explicit
Dim rs

Set rs = Server.CreateObject("ADODB.Recordset")
rs.Source = "SELECT id, name FROM users"
Response.Write CStr(rs.Source)

Set rs = Nothing
%>
```

530 Recordset.Status Property

Returns or sets record status flags for the current row context.

530.1 Syntax

```
value = rs.Status
rs.Status = newStatus
```

530.2 Return Value

Integer. Returns record status bit flags.

530.3 Remarks

- Property names are case-insensitive.
- Status values are provider/runtime dependent.
- Use for advanced state inspection in update workflows.

530.4 Code Example

```
<%
Option Explicit
Dim conn, rs

Set conn = Server.CreateObject("ADODB.Connection")
```

```
conn.ConnectionString = "Driver={SQLite3};Data Source=" & Server.MapPath
("./db.sqlite")
conn.Open
Set rs = conn.Execute("SELECT id, name FROM users")

If Not rs.EOF Then Response.Write CStr(rs.Status)

rs.Close
conn.Close
Set rs = Nothing
Set conn = Nothing
%>
```

531 Recordset.ActiveConnection Property

Gets or sets the connection used by the recordset.

531.1 Syntax

```
Set connRef = rs.ActiveConnection
Set rs.ActiveConnection = conn
```

531.2 Return Value

Object. Returns an ADODB.Connection object reference.

531.3 Remarks

- Property names are case-insensitive.
- Use Set for object assignment.
- Set before opening recordsets when source does not embed connection context.

531.4 Code Example

```
<%
Option Explicit
Dim conn, rs, connRef

Set conn = Server.CreateObject("ADODB.Connection")
conn.ConnectionString = "Driver={SQLite3};Data Source=" & Server.MapPath
("./db.sqlite")
conn.Open

Set rs = Server.CreateObject("ADODB.Recordset")
Set rs.ActiveConnection = conn
rs.Open "SELECT id, name FROM users", conn
Set connRef = rs.ActiveConnection
```

```
Response.Write "State: " & CStr(connRef.State)

rs.Close
conn.Close
Set connRef = Nothing
Set rs = Nothing
Set conn = Nothing
%>
```

532 Recordset.CursorType Property

Gets or sets the cursor type used by the recordset.

532.1 Syntax

```
value = rs.CursorType
rs.CursorType = newValue
```

532.2 Return Value

Integer. Returns cursor type flag.

532.3 Remarks

- Property names are case-insensitive.
- Set this value before opening the recordset.
- Cursor type affects movement and feature availability.

532.4 Code Example

```
<%
Option Explicit
Dim rs

Set rs = Server.CreateObject("ADODB.Recordset")
rs.CursorType = 3
Response.Write CStr(rs.CursorType)

Set rs = Nothing
%>
```

533 Recordset.LockType Property

Gets or sets the lock strategy used for record updates.

533.1 Syntax

```
value = rs.LockType
rs.LockType = newLockType
```

533.2 Return Value

Integer. Returns lock mode flag.

533.3 Remarks

- Property names are case-insensitive.
- Set before opening the recordset.
- Lock strategy affects concurrency and update behavior.

533.4 Code Example

```
<%
Option Explicit
Dim rs

Set rs = Server.CreateObject("ADODB.Recordset")
rs.LockType = 3
Response.Write CStr(rs.LockType)

Set rs = Nothing
%>
```

534 Recordset.CursorLocation Property

Gets or sets cursor location mode for the recordset.

534.1 Syntax

```
value = rs.CursorLocation
rs.CursorLocation = newValue
```

534.2 Return Value

Integer. Returns cursor location flag.

534.3 Remarks

- Property names are case-insensitive.
- Typical values are server-side and client-side cursor constants.
- Set prior to opening the recordset.

534.4 Code Example

```
<%
Option Explicit
Dim rs

Set rs = Server.CreateObject("ADODB.Recordset")
rs.CursorLocation = 3
```

```
Response.Write CStr(rs.CursorLocation)

Set rs = Nothing
%>
```

535 Command.ActiveConnection Property

Gets or sets the Connection object used by the command.

535.1 Syntax

```
Set connRef = cmd.ActiveConnection
Set cmd.ActiveConnection = conn
```

535.2 Return Value

Object. Returns an ADODB.Connection object reference.

535.3 Remarks

- Property names are case-insensitive.
- Use `Set` when assigning or reading this object property.
- This property must be set before calling `Command.Execute`.
- Reassigning ActiveConnection changes the connection target for future executions.

535.4 Code Example

```
<%
Option Explicit
Dim conn, cmd, connRef

Set conn = Server.CreateObject("ADODB.Connection")
conn.ConnectionString = "Driver={SQLite3};Data Source=" & Server.MapPath
("./db.sqlite")
conn.Open

Set cmd = Server.CreateObject("ADODB.Command")
Set cmd.ActiveConnection = conn
Set connRef = cmd.ActiveConnection

Response.Write "Connection state: " & CStr(connRef.State)

conn.Close
Set connRef = Nothing
Set cmd = Nothing
Set conn = Nothing
%>
```

536 Command.CommandText Property

Gets or sets the SQL statement or stored procedure name executed by the command.

536.1 Syntax

```
sql = cmd.CommandText
cmd.CommandText = "SELECT id, name FROM users"
```

536.2 Return Value

String. Returns the current command text.

536.3 Remarks

- Property names are case-insensitive.
- Set CommandText before calling Execute.
- For parameterized statements, use placeholders and append parameters in the same order.
- CommandText can contain SELECT, INSERT, UPDATE, DELETE, or provider-supported statements.

536.4 Code Example

```
<%
Option Explicit
Dim conn, cmd

Set conn = Server.CreateObject("ADODB.Connection")
conn.ConnectionString = "Driver={SQLite3};Data Source=" & Server.MapPath
("./db.sqlite")
conn.Open

Set cmd = Server.CreateObject("ADODB.Command")
Set cmd.ActiveConnection = conn
cmd.CommandText = "SELECT id, name FROM users WHERE active = 1"

Response.Write "Command text: " & cmd.CommandText

conn.Close
Set cmd = Nothing
Set conn = Nothing
%>
```

537 Command.CommandType Property

Gets or sets how CommandText should be interpreted.

537.1 Syntax

```
kind = cmd.CommandType
cmd.CommandType = 1
```

537.2 Return Value

Integer. Returns the command type flag.

537.3 Remarks

- Property names are case-insensitive.
- Common values are 1 for text command and 4 for stored procedure.
- Use 1 for SQL text in most AxonASP ADODB scenarios.
- Set CommandType before Execute for predictable provider behavior.

537.4 Code Example

```
<%
Option Explicit
Dim conn, cmd

Set conn = Server.CreateObject("ADODB.Connection")
conn.ConnectionString = "Driver={SQLite3};Data Source=" & Server.MapPath
("./db.sqlite")
conn.Open

Set cmd = Server.CreateObject("ADODB.Command")
Set cmd.ActiveConnection = conn
cmd.CommandType = 1
cmd.CommandText = "SELECT COUNT(*) AS total FROM users"

Response.Write "CommandType: " & CStr(cmd.CommandType)

conn.Close
Set cmd = Nothing
Set conn = Nothing
%>
```

538 Command.CommandTimeout Property

Gets or sets the maximum execution time for the command, in seconds.

538.1 Syntax

```
seconds = cmd.CommandTimeout
cmd.CommandTimeout = 30
```

538.2 Return Value

Integer. Returns the configured timeout value in seconds.

538.3 Remarks

- Property names are case-insensitive.
- Use this property to prevent long-running commands from blocking request execution.
- Set before calling Execute.
- Provider behavior can vary when timeout is `0` (infinite or provider default).

538.4 Code Example

```
<%
Option Explicit
Dim conn, cmd

Set conn = Server.CreateObject("ADODB.Connection")
conn.ConnectionString = "Driver={SQLite3};Data Source=" & Server.MapPath
("./db.sqlite")
conn.Open

Set cmd = Server.CreateObject("ADODB.Command")
Set cmd.ActiveConnection = conn
cmd.CommandText = "SELECT id FROM users"
cmd.CommandTimeout = 15

Response.Write "Timeout: " & CStr(cmd.CommandTimeout) & " seconds"

conn.Close
Set cmd = Nothing
Set conn = Nothing
%>
```

539 Command.Prepared Property

Gets or sets whether the command should be prepared before execution.

539.1 Syntax

```
isPrepared = cmd.Prepared
cmd.Prepared = True
```

539.2 Return Value

Boolean. Returns True when prepared mode is enabled; otherwise False.

539.3 Remarks

- Property names are case-insensitive.
- Prepared mode can improve performance for repeated command execution.
- Some providers may ignore this hint.
- Set this property before the first Execute call.

539.4 Code Example

```
<%
Option Explicit
Dim conn, cmd

Set conn = Server.CreateObject("ADODB.Connection")
conn.ConnectionString = "Driver={SQLite3};Data Source=" & Server.MapPath
("./db.sqlite")
conn.Open

Set cmd = Server.CreateObject("ADODB.Command")
Set cmd.ActiveConnection = conn
cmd.CommandText = "SELECT id, name FROM users WHERE active = 1"
cmd.Prepared = True

Response.Write "Prepared: " & CStr(cmd.Prepared)

conn.Close
Set cmd = Nothing
Set conn = Nothing
%>
```

540 Command.Parameters Property

Returns the Parameters collection bound to the command.

540.1 Syntax

```
Set params = cmd.Parameters
```

540.2 Return Value

Object. Returns an ADODB.Parameters collection object.

540.3 Remarks

- Property names are case-insensitive.
- This property is read-only.
- Use `CreateParameter` and `Parameters.Append` to populate the collection.

540.4 Code Example

```
<%
Option Explicit
Dim conn, cmd, params

Set conn = Server.CreateObject("ADODB.Connection")
conn.ConnectionString = "Driver={SQLite3};Data Source=" & Server.MapPath
("./db.sqlite")
conn.Open

Set cmd = Server.CreateObject("ADODB.Command")
Set cmd.ActiveConnection = conn
Set params = cmd.Parameters

Response.Write "Parameter count: " & CStr(params.Count)

conn.Close
Set params = Nothing
Set cmd = Nothing
Set conn = Nothing
%>
```

541 Error.Number Property

Returns the numeric error code for an ADODB error entry.

541.1 Syntax

```
code = errObj.Number
```

541.2 Return Value

Integer. Returns the provider or runtime-specific error number.

541.3 Remarks

- Property names are case-insensitive.
- This property is read-only.
- Use Number together with Description for diagnostics.
- Values are typically aligned with provider HRESULT-style errors.

541.4 Code Example

```
<%
Option Explicit
Dim conn, errObj

Set conn = Server.CreateObject("ADODB.Connection")
conn.ConnectionString = "Driver={SQLite3};Data Source=" & Server.MapPath
```

```
("./db.sqlite")
conn.Open

On Error Resume Next
conn.Execute "SELECT * FROM table_that_does_not_exist"
On Error GoTo 0

If conn.Errors.Count > 0 Then
    Set errObj = conn.Errors.Item(0)
    Response.Write "Error number: " & CStr(errObj.Number)
End If

conn.Close
Set errObj = Nothing
Set conn = Nothing
%>
```

542 Error.Description Property

Returns the human-readable message text for an ADODB error entry.

542.1 Syntax

```
text = errObj.Description
```

542.2 Return Value

String. Returns the error description message.

542.3 Remarks

- Property names are case-insensitive.
- This property is read-only.
- Use Description in logs or user-safe diagnostics.
- Pair with Number and SQLState for complete error context.

542.4 Code Example

```
<%
Option Explicit
Dim conn, errObj

Set conn = Server.CreateObject("ADODB.Connection")
conn.ConnectionString = "Driver={SQLite3};Data Source=" & Server.MapPath
("./db.sqlite")
conn.Open

On Error Resume Next
conn.Execute "SELECT * FROM table_that_does_not_exist"
```

```
On Error GoTo 0

If conn.Errors.Count > 0 Then
    Set errObj = conn.Errors.Item(0)
    Response.Write "Error description: " & errObj.Description
End If

conn.Close
Set errObj = Nothing
Set conn = Nothing
%>
```

543 Error.Source Property

Returns the component or method name that raised the ADODB error.

543.1 Syntax

```
name = errObj.Source
```

543.2 Return Value

String. Returns the error source text.

543.3 Remarks

- Property names are case-insensitive.
- This property is read-only.
- Source can help identify whether the fault came from SQL, provider, or command routing.
- Combine with Number and Description for full diagnostics.

543.4 Code Example

```
<%
Option Explicit
Dim conn, errObj

Set conn = Server.CreateObject("ADODB.Connection")
conn.ConnectionString = "Driver={SQLite3};Data Source=" & Server.MapPath
("./db.sqlite")
conn.Open

On Error Resume Next
conn.Execute "SELECT * FROM table_that_does_not_exist"
On Error GoTo 0

If conn.Errors.Count > 0 Then
    Set errObj = conn.Errors.Item(0)
```

```
    Response.Write "Error source: " & errObj.Source
End If

conn.Close
Set errObj = Nothing
Set conn = Nothing
%>
```

544 Error.SQLState Property

Returns the SQLSTATE code reported by the data provider.

544.1 Syntax

```
state = errObj.SQLState
```

544.2 Return Value

String. Returns the SQLSTATE code when available; can be an empty string for provider errors without SQLSTATE.

544.3 Remarks

- Property names are case-insensitive.
- This property is read-only.
- SQLState helps classify errors in a provider-neutral format.
- Not all providers populate SQLState for every error type.

544.4 Code Example

```
<%
Option Explicit
Dim conn, errObj

Set conn = Server.CreateObject("ADODB.Connection")
conn.ConnectionString = "Driver={SQLite3};Data Source=" & Server.MapPath
("./db.sqlite")
conn.Open

On Error Resume Next
conn.Execute "SELECT * FROM table_that_does_not_exist"
On Error GoTo 0

If conn.Errors.Count > 0 Then
    Set errObj = conn.Errors.Item(0)
    Response.Write "SQLState: " & errObj.SQLState
End If

conn.Close
```

```
Set errObj = Nothing
Set conn = Nothing
%>
```

545 Field.Value Property

Gets or sets the current value of a column in the active row.

545.1 Syntax

```
value = rs.Fields("columnName").Value
rs.Fields("columnName").Value = newValue
```

545.2 Return Value

Variant. Returns the current field value for the active row. Returns `Null` when the database value is null.

545.3 Remarks

- Property names are case-insensitive.
- Assigning to `Field.Value` marks the current row as edited.
- Call `Recordset.Update` to persist assigned values.
- Type conversion follows provider and ADODB coercion rules.

545.4 Code Example

```
<%
Option Explicit
Dim conn, rs

Set conn = Server.CreateObject("ADODB.Connection")
conn.ConnectionString = "Driver={SQLite3};Data Source=" & Server.MapPath
("./db.sqlite")
conn.Open

Set rs = conn.Execute("SELECT * FROM users WHERE id = 1")
If Not rs.EOF Then
    Response.Write "Current name: " & rs.Fields("name").Value & "<br>"
    rs.Fields("name").Value = "Updated Name"
    rs.Update
    Response.Write "Name updated"
End If

rs.Close
conn.Close
Set rs = Nothing
Set conn = Nothing
%>
```

546 Field.Name Property

Returns the column name represented by the Field object.

546.1 Syntax

```
name = rs.Fields(index).Name
```

546.2 Return Value

String. Returns the field name from the recordset schema.

546.3 Remarks

- Property names are case-insensitive.
- This property is read-only.
- Name reflects the alias when SQL uses AS.
- Use Name to build dynamic table headers and field maps.

546.4 Code Example

```
<%
Option Explicit
Dim conn, rs, i

Set conn = Server.CreateObject("ADODB.Connection")
conn.ConnectionString = "Driver={SQLite3};Data Source=" & Server.MapPath
("./db.sqlite")
conn.Open

Set rs = conn.Execute("SELECT id, name, email FROM users")
For i = 0 To rs.Fields.Count - 1
    Response.Write "Field " & CStr(i) & ": " & rs.Fields(i).Name & "<br>"
Next

rs.Close
conn.Close
Set rs = Nothing
Set conn = Nothing
%>
```

547 Field.Type Property

Returns the ADODB data type code for the field.

547.1 Syntax

```
typeCode = rs.Fields("columnName").Type
```

547.2 Return Value

Integer. Returns the ADODB type constant for the column.

547.3 Remarks

- Property names are case-insensitive.
- This property is read-only.
- Use Type to apply safe conversion logic when processing dynamic schemas.
- Type values depend on provider mappings.

547.4 Code Example

```
<%
Option Explicit
Dim conn, rs

Set conn = Server.CreateObject("ADODB.Connection")
conn.ConnectionString = "Driver={SQLite3};Data Source=" & Server.MapPath
("./db.sqlite")
conn.Open

Set rs = conn.Execute("SELECT id, name FROM users")
If Not rs.EOF Then
    Response.Write "id type: " & CStr(rs.Fields("id").Type) & "<br>"
    Response.Write "name type: " & CStr(rs.Fields("name").Type)
End If

rs.Close
conn.Close
Set rs = Nothing
Set conn = Nothing
%>
```

548 Field.DefinedSize Property

Returns the declared maximum size for the field in the schema.

548.1 Syntax

```
size = rs.Fields("columnName").DefinedSize
```

548.2 Return Value

Integer. Returns the column size declared by schema metadata.

548.3 Remarks

- Property names are case-insensitive.

- This property is read-only.
- DefinedSize is metadata; it does not depend on current row value.
- Compare with ActualSize to evaluate truncation risk.

548.4 Code Example

```
<%
Option Explicit
Dim conn, rs

Set conn = Server.CreateObject("ADODB.Connection")
conn.ConnectionString = "Driver={SQLite3};Data Source=" & Server.MapPath
("./db.sqlite")
conn.Open

Set rs = conn.Execute("SELECT name FROM users")
If Not rs.EOF Then
    Response.Write "Defined size: " & CStr(rs.Fields("name").DefinedSize)
End If

rs.Close
conn.Close
Set rs = Nothing
Set conn = Nothing
%>
```

549 Field.Attributes Property

Returns the attribute flags for the field metadata.

549.1 Syntax

```
flags = rs.Fields("columnName").Attributes
```

549.2 Return Value

Integer. Returns a bitmask describing field capabilities and metadata flags.

549.3 Remarks

- Property names are case-insensitive.
- This property is read-only.
- Flag values are provider-dependent.
- Common uses include checking nullable and auto-increment behavior.

549.4 Code Example

```
<%
Option Explicit
Dim conn, rs
```

```
Set conn = Server.CreateObject("ADODB.Connection")
conn.ConnectionString = "Driver={SQLite3};Data Source=" & Server.MapPath
("./db.sqlite")
conn.Open

Set rs = conn.Execute("SELECT id, name FROM users")
If Not rs.EOF Then
    Response.Write "Field attributes: " & CStr(rs.Fields("id").Attributes)
End If

rs.Close
conn.Close
Set rs = Nothing
Set conn = Nothing
%>
```

550 Field.NumericScale Property

Gets or sets the number of decimal places for numeric field values.

550.1 Syntax

```
scale = rs.Fields("amount").NumericScale
rs.Fields("amount").NumericScale = 2
```

550.2 Return Value

Integer. Returns the numeric scale for the field.

550.3 Remarks

- Property names are case-insensitive.
- Some providers expose this as read-only metadata.
- When writable, set before persisting values that require precision control.
- For non-numeric fields, value can be zero or provider-specific.

550.4 Code Example

```
<%
Option Explicit
Dim conn, rs

Set conn = Server.CreateObject("ADODB.Connection")
conn.ConnectionString = "Driver={SQLite3};Data Source=" & Server.MapPath
("./db.sqlite")
conn.Open

Set rs = conn.Execute("SELECT price FROM products")
```

```
If Not rs.EOF Then
    Response.Write "Numeric scale: " & CStr(rs.Fields("price").NumericSca
le)
End If

rs.Close
conn.Close
Set rs = Nothing
Set conn = Nothing
%>
```

551 Field.ActualSize Property

Returns the actual size of the current field value.

551.1 Syntax

```
length = rs.Fields("columnName").ActualSize
```

551.2 Return Value

Integer. Returns the current value length in characters for text data, or byte-length semantics as provided by the driver for binary-like values.

551.3 Remarks

- Property names are case-insensitive.
- This property is read-only.
- ActualSize reflects the current row value, not the schema maximum.
- Use DefinedSize for declared column capacity.

551.4 Code Example

```
<%
Option Explicit
Dim conn, rs

Set conn = Server.CreateObject("ADODB.Connection")
conn.ConnectionString = "Driver={SQLite3};Data Source=" & Server.MapPath
("./db.sqlite")
conn.Open

Set rs = conn.Execute("SELECT name FROM users WHERE id = 1")
If Not rs.EOF Then
    Response.Write "Actual size: " & CStr(rs.Fields("name").ActualSize)
End If

rs.Close
conn.Close
```

```
Set rs = Nothing
Set conn = Nothing
%>
```

552 Field.DataFormat Property

Returns the data format object associated with the field.

552.1 Syntax

```
fmt = rs.Fields("columnName").DataFormat
```

552.2 Return Value

Empty. In the current G3Pix AxonASP ADODB runtime, this property returns Empty.

552.3 Remarks

- Property names are case-insensitive.
- This property is exposed for compatibility.
- Assignments are not supported in the current implementation.

552.4 Code Example

```
<%
Option Explicit
Dim conn, rs, fmt

Set conn = Server.CreateObject("ADODB.Connection")
conn.ConnectionString = "Driver={SQLite3};Data Source=" & Server.MapPath
("./db.sqlite")
conn.Open

Set rs = conn.Execute("SELECT name FROM users")
If Not rs.EOF Then
    fmt = rs.Fields("name").DataFormat
    Response.Write "IsEmpty(DataFormat): " & CStr(IsEmpty(fmt))
End If

rs.Close
conn.Close
Set rs = Nothing
Set conn = Nothing
%>
```

553 Field.OriginalValue Property

Returns the original field value before local edits on the current row.

553.1 Syntax

```
value = rs.Fields("columnName").OriginalValue
```

553.2 Return Value

Variant. Returns the original value tracked by the runtime for the current row field.

553.3 Remarks

- Property names are case-insensitive.
- This property is read-only.
- In providers without full change tracking, value can match `Field.Value`.

553.4 Code Example

```
<%
Option Explicit
Dim conn, rs

Set conn = Server.CreateObject("ADODB.Connection")
conn.ConnectionString = "Driver={SQLite3};Data Source=" & Server.MapPath
("./db.sqlite")
conn.Open

Set rs = conn.Execute("SELECT * FROM users WHERE id = 1")
If Not rs.EOF Then
    Response.Write "Original: " & rs.Fields("name").OriginalValue & "<br>
"
    rs.Fields("name").Value = "Temporary Change"
    Response.Write "Current: " & rs.Fields("name").Value
End If

rs.Close
conn.Close
Set rs = Nothing
Set conn = Nothing
%>
```

554 Field.Precision Property

Returns numeric precision metadata for the field.

554.1 Syntax

```
value = rs.Fields("columnName").Precision
```

554.2 Return Value

Integer. Returns declared precision for numeric columns; provider-specific fallback may apply.

554.3 Remarks

- Property names are case-insensitive.
- This property is read-only.
- Use with `NumericScale` to validate decimal handling.

554.4 Code Example

```
<%
Option Explicit
Dim conn, rs

Set conn = Server.CreateObject("ADODB.Connection")
conn.ConnectionString = "Driver={SQLite3};Data Source=" & Server.MapPath
("./db.sqlite")
conn.Open
Set rs = conn.Execute("SELECT price FROM products")

If Not rs.EOF Then Response.Write CStr(rs.Fields("price").Precision)

rs.Close
conn.Close
Set rs = Nothing
Set conn = Nothing
%>
```

555 Field.Status Property

Returns status flags for the current field value.

555.1 Syntax

```
value = rs.Fields("columnName").Status
```

555.2 Return Value

Integer. Returns status information for the field according to provider/runtime flags.

555.3 Remarks

- Property names are case-insensitive.
- This property is read-only.
- Field status often reflects row update and null-state metadata.

555.4 Code Example

```
<%
Option Explicit
Dim conn, rs

Set conn = Server.CreateObject("ADODB.Connection")
conn.ConnectionString = "Driver={SQLite3};Data Source=" & Server.MapPath
("./db.sqlite")
conn.Open
Set rs = conn.Execute("SELECT id, name FROM users")

If Not rs.EOF Then Response.Write CStr(rs.Fields("name").Status)

rs.Close
conn.Close
Set rs = Nothing
Set conn = Nothing
%>
```

556 Field.UnderlyingValue Property

Returns the underlying provider value for the current field.

556.1 Syntax

```
value = rs.Fields("columnName").UnderlyingValue
```

556.2 Return Value

Variant. Returns the provider-level underlying value for the current row field.

556.3 Remarks

- Property names are case-insensitive.
- This property is read-only.
- For providers without separate change-tracking layers, value can match `Field.Value`.

556.4 Code Example

```
<%
Option Explicit
Dim conn, rs

Set conn = Server.CreateObject("ADODB.Connection")
conn.ConnectionString = "Driver={SQLite3};Data Source=" & Server.MapPath
("./db.sqlite")
conn.Open
Set rs = conn.Execute("SELECT name FROM users WHERE id = 1")
```

```
If Not rs.EOF Then Response.Write rs.Fields("name").UnderlyingValue

rs.Close
conn.Close
Set rs = Nothing
Set conn = Nothing
%>
```

557 Errors.Count Property

Returns the number of error entries stored in the connection Errors collection.

557.1 Syntax

```
count = conn.Errors.Count
```

557.2 Return Value

Integer. Returns the number of ADODB.Error items in the Errors collection.

557.3 Remarks

- Property names are case-insensitive.
- This property is read-only.
- The value is `0` when no errors are present.
- The value increases when operations fail and add entries to the Errors collection.

557.4 Code Example

```
<%
Option Explicit
Dim conn

Set conn = Server.CreateObject("ADODB.Connection")
conn.ConnectionString = "Driver={SQLite3};Data Source=" & Server.MapPath
("./db.sqlite")
conn.Open

On Error Resume Next
conn.Execute "SELECT * FROM table_that_does_not_exist"
On Error GoTo 0

Response.Write "Errors count: " & CStr(conn.Errors.Count)

conn.Close
Set conn = Nothing
%>
```

558 Errors.Item Property

Returns an error object from the Errors collection by index.

558.1 Syntax

```
Set errObj = conn.Errors.Item(index)
```

558.2 Return Value

Object or Empty. Returns an ADODB.Error object for a valid index, or Empty for out-of-range index access.

558.3 Remarks

- Property names are case-insensitive.
- Use `Errors.Count` before reading an item.
- Error objects expose `Number`, `Description`, `Source`, and `SQLState`.

558.4 Code Example

```
<%
Option Explicit
Dim conn, errObj

Set conn = Server.CreateObject("ADODB.Connection")
conn.ConnectionString = "Driver={SQLite3};Data Source=" & Server.MapPath
("./db.sqlite")
conn.Open

On Error Resume Next
conn.Execute "SELECT * FROM table_that_does_not_exist"
On Error GoTo 0

If conn.Errors.Count > 0 Then
    Set errObj = conn.Errors.Item(0)
    Response.Write errObj.Description
End If

conn.Close
Set errObj = Nothing
Set conn = Nothing
%>
```

559 Fields.Count Property

Returns the number of fields in the current Fields collection.

559.1 Syntax

```
count = rs.Fields.Count
```

559.2 Return Value

Integer. Returns total field entries in the collection.

559.3 Remarks

- Property names are case-insensitive.
- This property is read-only.
- Use with indexed `Fields.Item(i)` loops.

559.4 Code Example

```
<%
Option Explicit
Dim conn, rs

Set conn = Server.CreateObject("ADODB.Connection")
conn.ConnectionString = "Driver={SQLite3};Data Source=" & Server.MapPath
("./db.sqlite")
conn.Open
Set rs = conn.Execute("SELECT id, name FROM users")

Response.Write CStr(rs.Fields.Count)

rs.Close
conn.Close
Set rs = Nothing
Set conn = Nothing
%>
```

560 Fields.Item Property

Returns a field object from the Fields collection by index or name.

560.1 Syntax

```
Set fld = rs.Fields.Item(indexOrName)
```

560.2 Return Value

Object. Returns an ADODB.Field object.

560.3 Remarks

- Property names are case-insensitive.
- Invalid references raise runtime errors.

- Prefer names for readability and indexes for loops.

560.4 Code Example

```
<%
Option Explicit
Dim conn, rs, fld

Set conn = Server.CreateObject("ADODB.Connection")
conn.ConnectionString = "Driver={SQLite3};Data Source=" & Server.MapPath
("./db.sqlite")
conn.Open
Set rs = conn.Execute("SELECT id, name FROM users")

Set fld = rs.Fields.Item("name")
Response.Write fld.Value

rs.Close
conn.Close
Set fld = Nothing
Set rs = Nothing
Set conn = Nothing
%>
```

561 Parameters.Count Property

Returns the number of parameters in the command Parameters collection.

561.1 Syntax

```
count = cmd.Parameters.Count
```

561.2 Return Value

Integer. Returns total appended parameter entries.

561.3 Remarks

- Property names are case-insensitive.
- This property is read-only.
- Count reflects only parameters added to the current command object.

561.4 Code Example

```
<%
Option Explicit
Dim conn, cmd, p

Set conn = Server.CreateObject("ADODB.Connection")
conn.ConnectionString = "Driver={SQLite3};Data Source=" & Server.MapPath
```

```
("./db.sqlite")
conn.Open

Set cmd = Server.CreateObject("ADODB.Command")
Set cmd.ActiveConnection = conn
Set p = cmd.CreateParameter("id", 3, 1, 4, 1)
cmd.Parameters.Append p

Response.Write CStr(cmd.Parameters.Count)

conn.Close
Set p = Nothing
Set cmd = Nothing
Set conn = Nothing
%>
```

562 Parameters.Item Property

Returns a parameter object from the command Parameters collection.

562.1 Syntax

```
Set param = cmd.Parameters.Item(indexOrName)
```

562.2 Return Value

Object or Empty. Returns ADODB.Parameter for valid lookups; can return Empty for unresolved access in compatibility paths.

562.3 Remarks

- Property names are case-insensitive.
- Validate `Parameters.Count` before indexed reads.
- Name resolution depends on how parameters were created.

562.4 Code Example

```
<%
Option Explicit
Dim conn, cmd, p, p2

Set conn = Server.CreateObject("ADODB.Connection")
conn.ConnectionString = "Driver={SQLite3};Data Source=" & Server.MapPath
("./db.sqlite")
conn.Open

Set cmd = Server.CreateObject("ADODB.Command")
Set cmd.ActiveConnection = conn
Set p = cmd.CreateParameter("id", 3, 1, 4, 1)
```

```
cmd.Parameters.Append p

Set p2 = cmd.Parameters.Item(0)
Response.Write CStr(p2.Value)

conn.Close
Set p2 = Nothing
Set p = Nothing
Set cmd = Nothing
Set conn = Nothing
%>
```

563 Use ADODB.Stream in G3Pix AxonASP

563.1 Overview

Use `ADODB.Stream` to read, write, copy, and persist binary or text buffers with a cursor-based API.

563.2 Prerequisites

- Create the stream with `Server.CreateObject("ADODB.Stream")`.
- Call `Open` before using read or write operations.
- Set `Type` and `Charset` before text operations when you need explicit encoding behavior.

563.3 Syntax

```
Set stm = Server.CreateObject("ADODB.Stream")
```

563.4 Return Value

- **Object**: Returns an `ADODB.Stream` object handle.

563.5 How It Works

- Stream state is `0` when closed and `1` when open.
- `Read` returns byte-oriented data encoded as a VBScript-compatible byte string.
- `ReadText` decodes using the current `Charset`.
- `SaveToFile` and `LoadFromFile` operate inside AxonASP path resolution rules.

563.6 Remarks

- Member names are case-insensitive.
- Unsupported stream members return `Empty`.
- `Flush` is implemented as a compatibility no-op.

563.7 Example

```
<%
Option Explicit
Dim stm, textValue

Set stm = Server.CreateObject("ADODB.Stream")
stm.Type = 2
stm.Charset = "utf-8"
stm.Open

stm.WriteText "Hello from G3Pix AxonASP"
stm.Position = 0
textValue = stm.ReadText()

Response.Write textValue

stm.Close
Set stm = Nothing
%>
```

564 ADODB.Stream Methods Reference

564.1 Overview

Use these methods to operate on text and binary stream buffers in G3Pix AxonASP.

564.2 Methods

Method	Syntax	Parameters	Return Value
Open	`stm.Open`	none	**Empty**
Close	`stm.Close`	none	**Empty**
Read	`data = stm.Read([numBytes])`	`numBytes As Integer` (Optional)	**String** containing binary byte-string data
ReadText	`text = stm.ReadText([numChars])`	`numChars As Integer` (Optional)	**String**
Write	`stm.Write data`	`data As String` (Required, treated as byte-string payload)	**Empty**
WriteText	`stm.WriteText text [, writeLine]`	`text As String` (Required), `writeLine As Integer` (Optional,	**Empty**

Method	Syntax	Parameters	Return Value
		1 appends line separator)	
LoadFromFile	`stm.LoadFromFile path`	`path As String` (Required)	**Empty**
SaveToFile	`stm.SaveToFile path [, options]`	`path As String` (Required), `options As Integer` (Optional; 1 create-new, 2 overwrite/default)	**Empty**
CopyTo	`stm.CopyTo destination [, count]`	`destination As ADODB.Stream` (Required), `count As Integer` (Optional)	**Empty**
Flush	`stm.Flush`	none	**Empty** (compatibility no-op)
SetEOS	`stm.SetEOS`	none	**Empty**
SkipLine	`stm.SkipLine`	none	**Empty**

564.3 How It Works

- `Read` and `ReadText` advance `Position` by the amount consumed.
- `CopyTo` copies from the current source position and advances source position.
- `SetEOS` truncates stream data at the current `Position`.

564.4 Remarks

- `ReadText` returns an empty string when the stream is closed or at end-of-stream.
- `Write` and `WriteText` do nothing when the stream is closed.
- Charset-aware text behavior follows the current `Charset` property.

564.5 Example

```
<%
Option Explicit
Dim sourceStm, targetStm

Set sourceStm = Server.CreateObject("ADODB.Stream")
Set targetStm = Server.CreateObject("ADODB.Stream")

sourceStm.Type = 2
```

```
sourceStm.Charset = "utf-8"
sourceStm.Open
sourceStm.WriteText "Line 1"
sourceStm.WriteText "Line 2", 1
sourceStm.Position = 0

targetStm.Type = 2
targetStm.Charset = "utf-8"
targetStm.Open
sourceStm.CopyTo targetStm

targetStm.Position = 0
Response.Write targetStm.ReadText()

sourceStm.Close
targetStm.Close
Set sourceStm = Nothing
Set targetStm = Nothing
%>
```

565 ADODB.Stream Properties Reference

565.1 Overview

Use these properties to control encoding, cursor location, and stream metadata.

565.2 Properties

Property	Access	Type	Description
Type	Read/Write	Integer	Stream type (`1` binary, `2` text).
Mode	Read/Write	Integer	Mode flag value.
State	Read	Integer	Stream state (`0` closed, `1` open).
Position	Read/Write	Integer	Current cursor position in bytes.
Size	Read	Integer	Current stream size in bytes.
Charset	Read/Write	String	Text encoding name used by `ReadText` and `WriteText`.
LineSeparator	Read/Write	Integer	Line separator flag used when

Property	Access	Type	Description
			`WriteText` is called with append-line option.
EOS	Read	Boolean	`True` when `Position >= Size`.

565.3 How It Works

- Assigning `Position` clamps values to the valid range `0` to `Size`.
- `Charset` defaults to `unicode` for new stream objects.
- `Size` changes after write, load, copy, or truncation operations.

565.4 Remarks

- Property names are case-insensitive.
- Assigning an empty `Charset` resets it to `utf-8` in the current runtime.
- `EOS` is computed from `Position` and `Size`, and cannot be assigned directly.

566 Open Method

566.1 Overview

Opens the stream and initializes its internal cursor so read and write operations can run.

566.2 Prerequisites

- Create the object with `Server.CreateObject("ADODB.Stream")`.
- Set `Type` and optional text settings such as `Charset` before opening the stream.

566.3 Syntax

```
stm.Open
```

566.4 Parameters

This method does not accept parameters.

566.5 Return Value

Empty. The method does not return a value.

566.6 How It Works

- Changes `State` from `0` (closed) to `1` (open).
- Resets `Position` to the beginning of the in-memory buffer.

566.7 Remarks

- Method names are case-insensitive in G3Pix AxonASP.
- Calling `Open` on an already open stream keeps the stream available for I/O.

566.8 Code Example

```
<%
Option Explicit
Dim stm

Set stm = Server.CreateObject("ADODB.Stream")
stm.Type = 2
stm.Charset = "utf-8"
stm.Open

Response.Write "State: " & CStr(stm.State)

stm.Close
Set stm = Nothing
%>
```

566.9 API Reference

- Object: ADODB.Stream
- Method: Open
- Arguments: none
- Returns: Empty

567 Close Method

567.1 Overview

Closes the stream and releases access to its current buffer.

567.2 Prerequisites

- Create the object with `Server.CreateObject("ADODB.Stream")`.
- Open the stream before calling `Close` when you need deterministic cleanup.

567.3 Syntax

```
stm.Close
```

567.4 Parameters

This method does not accept parameters.

567.5 Return Value

Empty. The method does not return a value.

567.6 How It Works

- Sets `State` to `0` (closed).
- Keeps object instance valid so you can call `Open` again.

567.7 Remarks

- Method names are case-insensitive in G3Pix AxonASP.
- Calling `Close` on an already closed stream is safe.

567.8 Code Example

```
<%
Option Explicit
Dim stm

Set stm = Server.CreateObject("ADODB.Stream")
stm.Type = 2
stm.Open
stm.WriteText "temporary"
stm.Close

Response.Write "State after close: " & CStr(stm.State)

Set stm = Nothing
%>
```

567.9 API Reference

- Object: ADODB.Stream
- Method: Close
- Arguments: none
- Returns: Empty

568 Read Method

568.1 Overview

Reads binary data from the current stream position.

568.2 Prerequisites

- Create the object with `Server.CreateObject("ADODB.Stream")`.
- Set `Type = 1` for binary mode.
- Open the stream and ensure data exists in the buffer.

568.3 Syntax

```
data = stm.Read([numBytes])
```

568.4 Parameters

Parameter	Type	Required	Description
`numBytes`	Integer	No	Number of bytes to read from the current `Position`. If omitted, reads all remaining bytes until end of stream.

568.5 Return Value

String. Returns a byte-string payload containing the bytes read. Returns an empty string when the stream is closed, at end of stream, or no bytes are available.

568.6 How It Works

- Reads from current `Position`.
- Advances `Position` by the number of bytes returned.

568.7 Remarks

- Method names are case-insensitive in G3Pix AxonASP.
- Use `ReadText` instead of `Read` when `Type = 2` and text decoding is required.

568.8 Code Example

```
<%
Option Explicit
Dim stm, bytesChunk

Set stm = Server.CreateObject("ADODB.Stream")
stm.Type = 1
stm.Open
stm.Write ChrB(1) & ChrB(2) & ChrB(3) & ChrB(4)
stm.Position = 0

bytesChunk = stm.Read(2)
Response.Write "Bytes read: " & CStr(LenB(bytesChunk))

stm.Close
Set stm = Nothing
%>
```

568.9 API Reference

- Object: ADODB.Stream
- Method: Read
- Arguments: `numBytes As Integer` (optional)
- Returns: String (byte-string payload)

569 ReadText Method

569.1 Overview

Reads text from the current stream position using the configured character encoding.

569.2 Prerequisites

- Create the object with `Server.CreateObject("ADODB.Stream")`.
- Set `Type = 2` for text mode.
- Set `Charset` if you need a specific encoding.
- Open the stream before reading.

569.3 Syntax

```
text = stm.ReadText([numChars])
```

569.4 Parameters

Parameter	Type	Required	Description
`numChars`	Integer	No	Number of characters to read. If omitted, reads all remaining text to end of stream.

569.5 Return Value

String. Returns decoded text from the current position. Returns an empty string when the stream is closed, at end of stream, or no text remains.

569.6 How It Works

- Decodes bytes according to `Charset`.
- Moves `Position` forward by consumed bytes.

569.7 Remarks

- Method names are case-insensitive in G3Pix AxonASP.
- Use `Read` for binary payloads.

569.8 Code Example

```
<%
Option Explicit
Dim stm, part

Set stm = Server.CreateObject("ADODB.Stream")
stm.Type = 2
stm.Charset = "utf-8"
stm.Open
stm.WriteText "G3Pix AxonASP Stream"
stm.Position = 0

part = stm.ReadText(5)
Response.Write part

stm.Close
Set stm = Nothing
%>
```

569.9 API Reference

- Object: ADODB.Stream
- Method: ReadText
- Arguments: `numChars As Integer` (optional)
- Returns: String

570 Write Method

570.1 Overview

Writes binary data at the current stream position.

570.2 Prerequisites

- Create the object with `Server.CreateObject("ADODB.Stream")`.
- Set `Type = 1` for binary mode.
- Open the stream before writing.

570.3 Syntax

```
stm.Write data
```

570.4 Parameters

Parameter	Type	Required	Description
`data`	String	Yes	Byte-string payload to write into the

Parameter	Type	Required	Description
			stream buffer.

570.5 Return Value

Empty. The method does not return a value.

570.6 How It Works

- Writes bytes starting at current `Position`.
- Advances `Position` by number of bytes written.
- Expands `Size` when writing beyond current end.

570.7 Remarks

- Method names are case-insensitive in G3Pix AxonASP.
- Use `WriteText` for text-mode operations.

570.8 Code Example

```
<%
Option Explicit
Dim stm

Set stm = Server.CreateObject("ADODB.Stream")
stm.Type = 1
stm.Open
stm.Write ChrB(&HDE) & ChrB(&HAD) & ChrB(&HBE) & ChrB(&HEF)

Response.Write "Size after write: " & CStr(stm.Size)

stm.Close
Set stm = Nothing
%>
```

570.9 API Reference

- Object: ADODB.Stream
- Method: Write
- Arguments: `data As String`
- Returns: Empty

571 WriteText Method

571.1 Overview

Writes text data to the stream buffer using the current `Charset`.

571.2 Prerequisites

- Create the object with `Server.CreateObject("ADODB.Stream")`.
- Set `Type = 2` for text mode.
- Open the stream before writing.

571.3 Syntax

```
stm.WriteText text [, writeLine]
```

571.4 Parameters

Parameter	Type	Required	Description
`text`	String	Yes	Text to write at current stream position.
`writeLine`	Integer	No	When `1`, appends the configured line separator after `text`. Any other value writes only `text`.

571.5 Return Value

Empty. The method does not return a value.

571.6 How It Works

- Encodes the input with `Charset`.
- Writes bytes at current `Position`.
- Advances `Position` by bytes written.

571.7 Remarks

- Method names are case-insensitive in G3Pix AxonASP.
- Use `Write` for binary payloads.

571.8 Code Example

```
<%
Option Explicit
Dim stm

Set stm = Server.CreateObject("ADODB.Stream")
stm.Type = 2
stm.Charset = "utf-8"
stm.LineSeparator = 10
```

```
stm.Open
stm.WriteText "Header", 1
stm.WriteText "Body"

Response.Write "Size: " & CStr(stm.Size)

stm.Close
Set stm = Nothing
%>
```

571.9 API Reference

- Object: ADODB.Stream
- Method: WriteText
- Arguments: `text As String`, `writeLine As Integer` (optional)
- Returns: Empty

572 LoadFromFile Method

572.1 Overview

Loads file content into the stream buffer.

572.2 Prerequisites

- Create the object with `Server.CreateObject("ADODB.Stream")`.
- Open the stream before loading.

572.3 Syntax

```
stm.LoadFromFile path
```

572.4 Parameters

Parameter	Type	Required	Description
`path`	String	Yes	Absolute or mapped file path to read.

572.5 Return Value

Empty. The method does not return a value.

572.6 How It Works

- Replaces current in-memory stream data with file bytes.
- Sets `Position` to `0` after loading.
- Updates `Size` to loaded byte length.

572.7 Remarks

- Method names are case-insensitive in G3Pix AxonASP.
- In text mode, later `ReadText` operations decode bytes using current `Charset`.

572.8 Code Example

```
<%
Option Explicit
Dim stm, filePath

filePath = Server.MapPath("./data/sample.txt")
Set stm = Server.CreateObject("ADODB.Stream")
stm.Type = 2
stm.Charset = "utf-8"
stm.Open
stm.LoadFromFile filePath

Response.Write "Loaded bytes: " & CStr(stm.Size)

stm.Close
Set stm = Nothing
%>
```

572.9 API Reference

- Object: ADODB.Stream
- Method: LoadFromFile
- Arguments: `path As String`
- Returns: Empty

573 SaveToFile Method

573.1 Overview

Persists the current stream buffer to disk.

573.2 Prerequisites

- Create the object with `Server.CreateObject("ADODB.Stream")`.
- Open and populate the stream before saving.

573.3 Syntax

```
stm.SaveToFile path [, options]
```

573.4 Parameters

Parameter	Type	Required	Description

Parameter	Type	Required	Description
`path`	String	Yes	Destination file path.
`options`	Integer	No	Save option flag. 1 creates a new file and fails if it exists. 2 overwrites existing file content. Default is 2.

573.5 Return Value

Empty. The method does not return a value.

573.6 How It Works

- Writes the entire stream buffer to the target file.
- Does not close the stream after saving.

573.7 Remarks

- Method names are case-insensitive in G3Pix AxonASP.
- Use binary mode for raw bytes and text mode for encoded text payloads.

573.8 Code Example

```
<%
Option Explicit
Dim stm, outPath

outPath = Server.MapPath("./output/report.txt")
Set stm = Server.CreateObject("ADODB.Stream")
stm.Type = 2
stm.Charset = "utf-8"
stm.Open
stm.WriteText "Export generated"
stm.SaveToFile outPath, 2

Response.Write "Saved to disk"

stm.Close
Set stm = Nothing
%>
```

573.9 API Reference

- Object: ADODB.Stream

- Method: SaveToFile
- Arguments: `path As String`, `options As Integer` (optional)
- Returns: Empty

574 CopyTo Method

574.1 Overview

Copies stream data from the current source position to another open ADODB.Stream object.

574.2 Prerequisites

- Create both objects with `Server.CreateObject("ADODB.Stream")`.
- Open source and destination streams.
- Ensure both streams use compatible `Type` and encoding settings.

574.3 Syntax

```
stm.CopyTo destination [, count]
```

574.4 Parameters

Parameter	Type	Required	Description
`destination`	Object	Yes	Destination ADODB.Stream object that receives copied bytes.
`count`	Integer	No	Number of bytes to copy. If omitted, copies all remaining bytes from source position.

574.5 Return Value

Empty. The method does not return a value.

574.6 How It Works

- Reads bytes from source `Position`.
- Appends copied bytes at destination current `Position`.
- Advances source `Position` by copied byte count.

574.7 Remarks

- Method names are case-insensitive in G3Pix AxonASP.
- The destination stream remains open and reusable after copy.

574.8 Code Example

```
<%
Option Explicit
Dim sourceStm, targetStm

Set sourceStm = Server.CreateObject("ADODB.Stream")
Set targetStm = Server.CreateObject("ADODB.Stream")

sourceStm.Type = 2
sourceStm.Charset = "utf-8"
sourceStm.Open
sourceStm.WriteText "source payload"
sourceStm.Position = 0

targetStm.Type = 2
targetStm.Charset = "utf-8"
targetStm.Open
sourceStm.CopyTo targetStm
targetStm.Position = 0

Response.Write targetStm.ReadText()

sourceStm.Close
targetStm.Close
Set sourceStm = Nothing
Set targetStm = Nothing
%>
```

574.9 API Reference

- Object: ADODB.Stream
- Method: CopyTo
- Arguments: `destination As ADODB.Stream`, `count As Integer` (optional)
- Returns: Empty

575 Flush Method

575.1 Overview

Commits pending stream writes in runtimes that buffer I/O.

575.2 Prerequisites

- Create the object with `Server.CreateObject("ADODB.Stream")`.
- Open the stream before calling `Flush`.

575.3 Syntax

```
stm.Flush
```

575.4 Parameters

This method does not accept parameters.

575.5 Return Value

Empty. The method does not return a value.

575.6 How It Works

- In current G3Pix AxonASP ADODB.Stream implementation, `Flush` is a compatibility no-op.
- The call does not modify `Position`, `Size`, or stream content.

575.7 Remarks

- Method names are case-insensitive in G3Pix AxonASP.
- Keep this call in legacy code paths that expect explicit flush points.

575.8 Code Example

```
<%
Option Explicit
Dim stm

Set stm = Server.CreateObject("ADODB.Stream")
stm.Type = 2
stm.Open
stm.WriteText "buffered text"
stm.Flush

Response.Write "Flush executed"

stm.Close
Set stm = Nothing
%>
```

575.9 API Reference

- Object: ADODB.Stream
- Method: Flush

- Arguments: none
- Returns: Empty

576 SetEOS Method

576.1 Overview

Truncates the stream so end-of-stream matches the current cursor position.

576.2 Prerequisites

- Create the object with `Server.CreateObject("ADODB.Stream")`.
- Open the stream before truncation.

576.3 Syntax

```
stm.SetEOS
```

576.4 Parameters

This method does not accept parameters.

576.5 Return Value

Empty. The method does not return a value.

576.6 How It Works

- Removes all bytes after current `Position`.
- Updates `Size` to the new truncated length.

576.7 Remarks

- Method names are case-insensitive in G3Pix AxonASP.
- Use this method to discard trailing buffer content after partial edits.

576.8 Code Example

```
<%
Option Explicit
Dim stm

Set stm = Server.CreateObject("ADODB.Stream")
stm.Type = 2
stm.Open
stm.WriteText "abcdef"
stm.Position = 3
stm.SetEOS
stm.Position = 0
```

```
Response.Write stm.ReadText()

stm.Close
Set stm = Nothing
%>
```

576.9 API Reference

- Object: ADODB.Stream
- Method: SetEOS
- Arguments: none
- Returns: Empty

577 SkipLine Method

577.1 Overview

Moves the stream position to the beginning of the next text line.

577.2 Prerequisites

- Create the object with `Server.CreateObject("ADODB.Stream")`.
- Use text mode (`Type = 2`) and open the stream.

577.3 Syntax

```
stm.SkipLine
```

577.4 Parameters

This method does not accept parameters.

577.5 Return Value

Empty. The method does not return a value.

577.6 How It Works

- Scans from current `Position` until it finds line separator bytes.
- Sets `Position` to the first character after the separator.

577.7 Remarks

- Method names are case-insensitive in G3Pix AxonASP.
- In binary mode, line-based navigation is not meaningful.

577.8 Code Example

```
<%
Option Explicit
Dim stm
```

```
Set stm = Server.CreateObject("ADODB.Stream")
stm.Type = 2
stm.Charset = "utf-8"
stm.LineSeparator = 10
stm.Open
stm.WriteText "first", 1
stm.WriteText "second", 1
stm.Position = 0
stm.SkipLine

Response.Write stm.ReadText(6)

stm.Close
Set stm = Nothing
%>
```

577.9 API Reference

- Object: ADODB.Stream
- Method: SkipLine
- Arguments: none
- Returns: Empty

578 Type Property

578.1 Overview

Gets or sets the stream content mode.

578.2 Syntax

```
value = stm.Type
stm.Type = newValue
```

578.3 Return Value

Integer. Returns `1` for binary mode or `2` for text mode.

578.4 Remarks

- Property names are case-insensitive in G3Pix AxonASP.
- Set `Type = 1` before binary `Read` and `Write` operations.
- Set `Type = 2` before `ReadText` and `WriteText` operations.

578.5 Code Example

```
<%
Option Explicit
Dim stm
```

```
Set stm = Server.CreateObject("ADODB.Stream")
stm.Type = 2
stm.Open
stm.WriteText "text payload"
Response.Write "Type: " & CStr(stm.Type)

stm.Close
Set stm = Nothing
%>
```

578.6 API Reference

- Object: ADODB.Stream
- Property: Type
- Access: Read/Write
- Type: Integer (1 binary, 2 text)

579 Mode Property

579.1 Overview

Gets or sets stream access mode flags.

579.2 Syntax

```
value = stm.Mode
stm.Mode = newValue
```

579.3 Return Value

Integer. Returns the current mode flag value.

579.4 Remarks

- Property names are case-insensitive in G3Pix AxonASP.
- This property is exposed for ADODB compatibility and can be assigned before opening the stream.
- Runtime enforcement of specific lock/share semantics can vary by host environment.

579.5 Code Example

```
<%
Option Explicit
Dim stm

Set stm = Server.CreateObject("ADODB.Stream")
stm.Mode = 3
```

```
stm.Open

Response.Write "Mode: " & CStr(stm.Mode)

stm.Close
Set stm = Nothing
%>
```

579.6 API Reference

- Object: ADODB.Stream
- Property: Mode
- Access: Read/Write
- Type: Integer

580 State Property

580.1 Overview

Returns whether the stream is open or closed.

580.2 Syntax

```
value = stm.State
```

580.3 Return Value

Integer. Returns `0` when the stream is closed and `1` when the stream is open.

580.4 Remarks

- Property names are case-insensitive in G3Pix AxonASP.
- This property is read-only.
- Check `State` before calling read and write methods.

580.5 Code Example

```
<%
Option Explicit
Dim stm

Set stm = Server.CreateObject("ADODB.Stream")
Response.Write "Before open: " & CStr(stm.State) & "<br>"

stm.Open
Response.Write "After open: " & CStr(stm.State) & "<br>"

stm.Close
Response.Write "After close: " & CStr(stm.State)
```

```
Set stm = Nothing
%>
```

580.6 API Reference

- Object: ADODB.Stream
- Property: State
- Access: Read-only
- Type: Integer (`0` closed, `1` open)

581 Position Property

581.1 Overview

Gets or sets the current cursor position in the stream buffer.

581.2 Syntax

```
value = stm.Position
stm.Position = newValue
```

581.3 Return Value

Integer. Returns the current byte offset from the start of the stream.

581.4 Remarks

- Property names are case-insensitive in G3Pix AxonASP.
- Valid range is `0` to `Size`.
- Assignments below `0` clamp to `0`; assignments above `Size` clamp to `Size`.

581.5 Code Example

```
<%
Option Explicit
Dim stm

Set stm = Server.CreateObject("ADODB.Stream")
stm.Type = 2
stm.Open
stm.WriteText "abcdef"
stm.Position = 2

Response.Write "Current position: " & CStr(stm.Position)

stm.Close
Set stm = Nothing
%>
```

581.6 API Reference

- Object: ADODB.Stream
- Property: Position
- Access: Read/Write
- Type: Integer

582 Size Property

582.1 Overview

Returns the current stream length in bytes.

582.2 Syntax

```
value = stm.Size
```

582.3 Return Value

Integer. Returns total byte count stored in the stream buffer.

582.4 Remarks

- Property names are case-insensitive in G3Pix AxonASP.
- This property is read-only.
- `Size` changes after `Write`, `WriteText`, `LoadFromFile`, `CopyTo`, or `SetEOS`.

582.5 Code Example

```
<%
Option Explicit
Dim stm

Set stm = Server.CreateObject("ADODB.Stream")
stm.Type = 2
stm.Open
stm.WriteText "hello"

Response.Write "Size in bytes: " & CStr(stm.Size)

stm.Close
Set stm = Nothing
%>
```

582.6 API Reference

- Object: ADODB.Stream
- Property: Size
- Access: Read-only

- Type: Integer

583 Charset Property

583.1 Overview

Gets or sets the text encoding used by `ReadText` and `WriteText`.

583.2 Syntax

```
value = stm.Charset
stm.Charset = newValue
```

583.3 Return Value

String. Returns the active charset name.

583.4 Remarks

- Property names are case-insensitive in G3Pix AxonASP.
- This property affects text mode (`Type = 2`) operations.
- New stream objects default to `unicode`.
- Assigning an empty charset value resets to `utf-8` in current runtime behavior.

583.5 Code Example

```
<%
Option Explicit
Dim stm

Set stm = Server.CreateObject("ADODB.Stream")
stm.Type = 2
stm.Charset = "utf-8"
stm.Open
stm.WriteText "cafe"

Response.Write "Charset: " & stm.Charset

stm.Close
Set stm = Nothing
%>
```

583.6 API Reference

- Object: ADODB.Stream
- Property: Charset
- Access: Read/Write
- Type: String

584 LineSeparator Property

584.1 Overview

Gets or sets the line separator written by `WriteText` when append-line mode is enabled.

584.2 Syntax

```
value = stm.LineSeparator
stm.LineSeparator = newValue
```

584.3 Return Value

Integer. Returns the active line separator flag.

584.4 Remarks

- Property names are case-insensitive in G3Pix AxonASP.
- Common values are `10` (LF) and `13` (CR).
- This property is used only when `WriteText` is called with the optional append-line flag set to `1`.

584.5 Code Example

```
<%
Option Explicit
Dim stm

Set stm = Server.CreateObject("ADODB.Stream")
stm.Type = 2
stm.Charset = "utf-8"
stm.LineSeparator = 10
stm.Open
stm.WriteText "line1", 1
stm.WriteText "line2", 1

Response.Write "LineSeparator: " & CStr(stm.LineSeparator)

stm.Close
Set stm = Nothing
%>
```

584.6 API Reference

- Object: ADODB.Stream
- Property: LineSeparator
- Access: Read/Write
- Type: Integer

585 EOS Property

585.1 Overview

Indicates whether current stream position is at or beyond end-of-stream.

585.2 Syntax

```
value = stm.EOS
```

585.3 Return Value

Boolean. Returns `True` when `Position >= Size`; otherwise returns `False`.

585.4 Remarks

- Property names are case-insensitive in G3Pix AxonASP.
- This property is read-only.
- Use `EOS` in loops to stop reading safely.

585.5 Code Example

```
<%
Option Explicit
Dim stm

Set stm = Server.CreateObject("ADODB.Stream")
stm.Type = 2
stm.Open
stm.WriteText "abc"
stm.Position = 0

Do While Not stm.EOS
    Response.Write stm.ReadText(1)
Loop

stm.Close
Set stm = Nothing
%>
```

585.6 API Reference

- Object: ADODB.Stream
- Property: EOS
- Access: Read-only
- Type: Boolean

586 Use Scripting.Dictionary in G3Pix AxonASP

586.1 Overview

Use Scripting.Dictionary to store key-value pairs with ordered keys and fast key lookup.

586.2 Prerequisites

- Create the object with the primary ProgID Scripting.Dictionary.
- Use consistent key types when possible to keep lookups predictable.

586.3 Syntax

```
Set dict = Server.CreateObject("Scripting.Dictionary")
```

586.4 Parameters and Arguments

- ProgID (String, required): Scripting.Dictionary.

586.5 Return Value

`Server.CreateObject("Scripting.Dictionary")` returns an object reference to a new, empty dictionary.

586.6 How It Works

- Keys are preserved in insertion order for Keys and Items enumeration.
- CompareMode controls key normalization:
  - `0` = BinaryCompare (case-sensitive)
  - `1` = TextCompare (case-insensitive)
- Reading a missing key through Item creates that key with Empty value.

586.7 Remarks

- Member names are case-insensitive.
- The object supports both property and method access patterns used by classic VBScript dictionary code.

586.8 Code Example

```
<%
Option Explicit

Dim dict
Set dict = Server.CreateObject("Scripting.Dictionary")

dict.Add "Name", "AxonASP"
dict.Item("Version") = "2"
```

```
Response.Write "Count=" & dict.Count & "<br>"
Response.Write "Name=" & dict.Item("Name") & "<br>"
Response.Write "Version=" & dict.Item("Version")

Set dict = Nothing
%>
```

587 Scripting.Dictionary Methods

587.1 Overview

This page summarizes methods exposed by Scripting.Dictionary in G3Pix AxonASP.

587.2 Methods Reference

Method	Returns	Description
Add(key, value)	Empty	Adds a new entry. Raises an error when the key already exists or required arguments are missing.
Exists(key)	Boolean or Empty	Returns True when the key exists, False when it does not. Returns Empty after argument-count error handling.
Remove(key)	Empty	Removes one key. Raises an error when the key does not exist or required arguments are missing.
RemoveAll()	Empty	Clears all keys and values.
Keys()	Array	Returns a zero-based Variant array containing keys in insertion order.
Items()	Array	Returns a zero-based Variant array containing values in insertion order.
Item(key [, value])	Value or Empty	With key only, returns the item value and auto-creates missing keys with Empty. With key and value, assigns and returns

Method	Returns	Description
		Empty.
Key(oldKey, newKey)	Empty	Renames an existing key. Raises an error when oldKey is missing, oldKey is not found, or newKey already exists.
Count()	Integer	Returns the number of dictionary entries.

587.3 Remarks

- Method names are case-insensitive.
- Item and Key also participate in assignment syntax patterns used by VBScript code.

588 Scripting.Dictionary Properties

588.1 Overview

This page summarizes properties exposed by Scripting.Dictionary in G3Pix AxonASP.

588.2 Properties Reference

Property	Access	Type	Description
Count	Read-only	Integer	Returns the number of key-value entries currently stored.
CompareMode	Read/Write	Integer	Gets or sets key comparison mode: 0 for case-sensitive comparison, 1 for case-insensitive comparison.

588.3 Remarks

- Property names are case-insensitive.
- Set CompareMode before adding entries. Changing CompareMode after the dictionary contains items raises an error.

589 Add a Dictionary Entry

589.1 Overview

Use Add to insert a new key-value pair into the dictionary.

589.2 Syntax

```
dict.Add key, value
```

589.3 Parameters

- key (Variant, required): Entry key.
- value (Variant, required): Entry value.

589.4 Return Value

Returns Empty.

589.5 How It Works

- Add inserts the key and value at the end of insertion order.
- If the key already exists, Add raises an error and does not overwrite the value.

589.6 Remarks

- Add requires exactly two arguments.
- Member names are case-insensitive.

589.7 Example

```
<%
Option Explicit

Dim dict
Set dict = Server.CreateObject("Scripting.Dictionary")

dict.Add "Language", "VBScript"
Response.Write dict.Item("Language")

Set dict = Nothing
%>
```

590 Count Dictionary Entries

590.1 Overview

Use Count to read how many entries are currently stored.

590.2 Syntax

```
countValue = dict.Count()
```

590.3 Parameters

- No parameters.

590.4 Return Value

Returns an Integer containing the number of entries.

590.5 How It Works

- Count reads the current key slice length.
- Add, Remove, RemoveAll, and Item assignment can change this value.

590.6 Remarks

- Count is also available as property-style access: dict.Count.
- Member names are case-insensitive.

590.7 Example

```
<%
Option Explicit

Dim dict
Set dict = Server.CreateObject("Scripting.Dictionary")

dict.Add "A", 1
dict.Add "B", 2

Response.Write dict.Count()

Set dict = Nothing
%>
```

591 Check Whether a Key Exists

591.1 Overview

Use Exists to test if a dictionary key is present.

591.2 Syntax

```
found = dict.Exists(key)
```

591.3 Parameters

- key (Variant, required): Key to test.

591.4 Return Value

Returns Boolean True when the key exists.

Returns Boolean False when the key does not exist.

591.5 How It Works

- Exists uses the dictionary index map and CompareMode rules.
- Missing argument raises an error in runtime validation.

591.6 Remarks

- Member names are case-insensitive.

591.7 Example

```
<%
Option Explicit

Dim dict
Set dict = Server.CreateObject("Scripting.Dictionary")

dict.Add "Region", "US"
Response.Write CStr(dict.Exists("Region"))

Set dict = Nothing
%>
```

592 Get or Set a Dictionary Item

592.1 Overview

Use Item to read or assign a value by key.

592.2 Syntax

```
value = dict.Item(key)
dict.Item(key) = value
```

592.3 Parameters

- key (Value, required): Entry key.
- value (Value, required in assignment form): Value assigned when using set form.

592.4 Return Value

With one argument, returns the stored value for key.

When key does not exist in get form, Item returns Empty and creates the key with Empty value.

With assignment form, Item returns Empty.

592.5 How It Works

- Lookup respects CompareMode.
- Assignment updates existing keys or creates new keys.

592.6 Remarks

- Member names are case-insensitive.
- The same behavior is available through default-index syntax: `dict(key)`.

592.7 Example

```
<%
Option Explicit

Dim dict, missingValue
Set dict = Server.CreateObject("Scripting.Dictionary")

dict.Item("Mode") = "Production"
missingValue = dict.Item("UndefinedKey")

Response.Write "Mode=" & dict.Item("Mode") & "<br>"
Response.Write "Missing is empty=" & CStr(IsEmpty(missingValue))

Set dict = Nothing
%>
```

593 Enumerate Dictionary Values

593.1 Overview

Use Items to retrieve all stored values as an array.

593.2 Syntax

```
values = dict.Items()
```

593.3 Parameters

- No parameters.

593.4 Return Value

Returns a zero-based Variant array containing dictionary values in insertion order.

593.5 How It Works

- The returned array length equals Count.
- Values are copied from the dictionary storage order.

593.6 Remarks

- Member names are case-insensitive.

593.7 Example

```
<%
Option Explicit

Dim dict, values, i
Set dict = Server.CreateObject("Scripting.Dictionary")

dict.Add "A", 10
dict.Add "B", 20
values = dict.Items()

For i = 0 To UBound(values)
    Response.Write "Value(" & i & ")=" & values(i) & "<br>"
Next

Set dict = Nothing
%>
```

594 Rename a Dictionary Key

594.1 Overview

Use Key to rename an existing dictionary key without changing its associated value.

594.2 Syntax

```
dict.Key(oldKey) = newKey
```

594.3 Parameters

- oldKey (Variant, required): Existing key to rename.
- newKey (Variant, required): New key name.

594.4 Return Value

Returns Empty.

594.5 How It Works

- The key position is preserved in insertion order.
- Renaming fails when oldKey does not exist.

- Renaming fails when newKey already exists.

594.6 Remarks

- Runtime assignment syntax is the recommended form.
- Member names are case-insensitive.

594.7 Example

```
<%
Option Explicit

Dim dict
Set dict = Server.CreateObject("Scripting.Dictionary")

dict.Add "OldName", "Value"
dict.Key("OldName") = "NewName"

Response.Write dict.Item("NewName")

Set dict = Nothing
%>
```

595 Enumerate Dictionary Keys

595.1 Overview

Use Keys to retrieve all dictionary keys as an array.

595.2 Syntax

```
keys = dict.Keys()
```

595.3 Parameters

- No parameters.

595.4 Return Value

Returns a zero-based Variant array containing keys in insertion order.

595.5 How It Works

- Keys are copied from dictionary internal key order.
- The returned array length equals Count.

595.6 Remarks

- Member names are case-insensitive.

595.7 Example

```
<%
Option Explicit

Dim dict, keys, i
Set dict = Server.CreateObject("Scripting.Dictionary")

dict.Add "A", "x"
dict.Add "B", "y"
keys = dict.Keys()

For i = 0 To UBound(keys)
    Response.Write "Key(" & i & ")=" & keys(i) & "<br>"
Next

Set dict = Nothing
%>
```

596 Remove a Dictionary Entry

596.1 Overview

Use Remove to delete one dictionary entry by key.

596.2 Syntax

```
dict.Remove key
```

596.3 Parameters

- key (Variant, required): Key to remove.

596.4 Return Value

Returns Empty.

596.5 How It Works

- Remove deletes both key and value and rebuilds the internal index map.
- Remove raises an error when the key does not exist.

596.6 Remarks

- Remove requires one argument.
- Member names are case-insensitive.

596.7 Example

```
<%
Option Explicit
```

```
Dim dict
Set dict = Server.CreateObject("Scripting.Dictionary")

dict.Add "Token", "ABC"
dict.Remove "Token"

Response.Write "Count=" & dict.Count

Set dict = Nothing
%>
```

597 Clear All Dictionary Entries

597.1 Overview

Use RemoveAll to clear all key-value entries in the dictionary.

597.2 Syntax

```
dict.RemoveAll
```

597.3 Parameters

- No parameters.

597.4 Return Value

Returns Empty.

597.5 How It Works

- RemoveAll clears key and value storage.
- Count becomes 0 after completion.

597.6 Remarks

- Member names are case-insensitive.

597.7 Example

```
<%
Option Explicit

Dim dict
Set dict = Server.CreateObject("Scripting.Dictionary")

dict.Add "A", 1
dict.Add "B", 2
dict.RemoveAll
```

```
Response.Write "Count=" & dict.Count

Set dict = Nothing
%>
```

598 Set Dictionary CompareMode

598.1 Overview

Use CompareMode to control how keys are compared during lookup and existence checks.

598.2 Syntax

```
mode = dict.CompareMode
dict.CompareMode = modeValue
```

598.3 Parameters

- modeValue (Integer, required for assignment):
  - `0` = BinaryCompare (case-sensitive)
  - `1` = TextCompare (case-insensitive)

598.4 Return Value

Getter returns an Integer representing the current comparison mode.

Setter returns no value.

598.5 How It Works

- CompareMode changes key normalization rules used by Exists, Item, Add, Remove, and Key.
- CompareMode can be changed only while the dictionary is empty.

598.6 Remarks

- Assigning CompareMode after entries already exist raises an error.
- Member names are case-insensitive.

598.7 Example

```
<%
Option Explicit

Dim dict
Set dict = Server.CreateObject("Scripting.Dictionary")

dict.CompareMode = 1
dict.Add "Token", "A"
```

```
Response.Write CStr(dict.Exists("token"))

Set dict = Nothing
%>
```

599 Get Dictionary Entry Count

599.1 Overview

Use Count to retrieve the current number of entries in a Scripting.Dictionary instance.

599.2 Syntax

```
countValue = dict.Count
countValue = dict.Count()
```

599.3 Parameters

- Getter only. This member accepts no arguments.

599.4 Return Value

Returns an Integer equal to the number of entries currently stored.

599.5 How It Works

- Count reflects the number of keys in the dictionary.
- The value updates immediately after Add, Remove, RemoveAll, and Item assignment that creates new keys.

599.6 Remarks

- Count is exposed both as property-style and method-style access.
- Member names are case-insensitive.

599.7 Example

```
<%
Option Explicit

Dim dict
Set dict = Server.CreateObject("Scripting.Dictionary")

dict.Add "A", 10
dict.Add "B", 20

Response.Write "Count=" & dict.Count
```

```
Set dict = Nothing
%>
```

600 Use Scripting.FileSystemObject in AxonASP

600.1 Overview

Scripting.FileSystemObject provides access to the server file system from Classic ASP scripts. Use it to create, read, update, and delete files and folders; navigate directory hierarchies; open text streams for reading or writing; and query drive information.

Creating the object returns a root FSO handle. From it you obtain **File**, **Folder**, **TextStream**, and **Drive** sub-objects, each exposing their own methods and properties.

600.2 Prerequisites

- The target path must reside within the server document root or a mapped physical path.
- Write operations require that the server process has the necessary file system permissions on the target path.
- All relative paths are resolved against the document root before being forwarded to the operating system.

600.3 Syntax

```
Set fso = Server.CreateObject("Scripting.FileSystemObject")
```

600.4 Return Value

Returns a native FSO object handle. Use `Set` for assignment. Call `Set fso = Nothing` when the object is no longer needed.

600.5 How It Works

The AxonASP runtime resolves all relative paths against the web document root and normalizes path separators for the host operating system before passing them to the OS file API. Closing a TextStream or assigning `Nothing` to a File or Folder handle releases the underlying OS file handle immediately.

The FSO object model is composed of several distinct sub-objects, each returned by methods on the root FSO:

- **File** — Represents a single file entry with metadata properties and stream access.
- **Folder** — Represents a directory with properties, sub-folder and file collections, and a text-file factory.
- **TextStream** — A sequential text I/O stream for reading from or writing to a file.

- **Drive** — Exposes capacity, type, and volume information for one drive.
- **DrivesCollection** — Enumerates all available drives on the server.

600.6 Remarks

- All method and property names are case-insensitive.
- On non-Windows hosts, drive-related properties return normalized values representing the root file system.
- `GetTempName` generates a unique temporary file name in `radXXXXXX.axon.tmp` format and does **not** create the file on disk.
- Attribute values follow classic FSO integer constants: `1` = ReadOnly, `2` = Hidden, `16` = Directory, `32` = Archive.

600.7 Code Example

```
<%
Option Explicit
Dim fso, ts, f

Set fso = Server.CreateObject("Scripting.FileSystemObject")

' Write a text file
Set ts = fso.CreateTextFile(Server.MapPath("data/hello.txt"), True)
ts.WriteLine "Hello, World!"
ts.Close
Set ts = Nothing

' Read it back
Set ts = fso.OpenTextFile(Server.MapPath("data/hello.txt"), 1)
Response.Write ts.ReadAll
ts.Close
Set ts = Nothing

' Inspect the file
Set f = fso.GetFile(Server.MapPath("data/hello.txt"))
Response.Write f.Size & " bytes, modified " & f.DateLastModified
Set f = Nothing

Set fso = Nothing
%>
```

601 Methods

601.1 Overview

This page lists all methods exposed by the root Scripting.FileSystemObject handle. Methods that return sub-objects (File, Folder, TextStream, Drive) require `Set` for assignment.

601.2 FileSystemObject Methods

Method	Returns	Description
BuildPath(path, name)	String	Concatenates a path and a file name using the OS path separator.
CopyFile(source, destination [, overwrite])	Empty	Copies one file to the destination path. Overwrites by default.
CopyFolder(source, destination [, overwrite])	Empty	Recursively copies a folder tree to the destination path. Overwrites by default.
CreateFolder(folderspec)	Folder	Creates the specified folder (and any missing parent folders) and returns a Folder object.
CreateTextFile(filename [, overwrite])	TextStream	Creates a new file opened for writing and returns a TextStream. If overwrite is False and the file exists, returns Empty.
DeleteFile(filespec)	Empty	Deletes the specified file. Raises error 53 (File not found) or error 70 (Permission denied) on failure.
DeleteFolder(folderspec)	Empty	Deletes the specified folder and all its contents. Raises error 76 (Path not found) or error 70 on failure.
DriveExists(drivespec)	Boolean	Returns True if the

Method	Returns	Description
		specified drive or drive path exists; False otherwise.
FileExists(filespec)	Boolean	Returns True if the specified file exists; False otherwise.
FolderExists(folderspec)	Boolean	Returns True if the specified folder exists; False otherwise.
GetAbsolutePathName(pathspec)	String	Returns the fully resolved absolute path for the given relative or partial path. Returns an empty String if the path cannot be resolved.
GetBaseName(pathspec)	String	Returns the base file name without its extension.
GetDrive(drivespec)	Drive	Returns a Drive object for the specified drive letter or path. Returns Empty if the drive name cannot be determined.
GetDriveName(pathspec)	String	Returns the drive letter or identifier extracted from a path. Returns an empty String if not applicable.
GetExtensionName(pathspec)	String	Returns the file extension without the leading period. Returns an empty String if there is no extension.
GetFile(filespec)	File	Returns a File object for the specified file path. Raises error 53 if the file does not exist.
GetFileName(pathspec)	String	Returns the last path component, including the extension. Returns an

Method	Returns	Description
		empty String if the path is empty.
GetFileVersion(filespec)	String	Returns the version resource string of the specified file. Returns “1.0.0.0” if no version information is available.
GetFolder(folderspec)	Folder	Returns a Folder object for the specified folder path. Raises error 76 if the folder does not exist.
GetParentFolderName(pathspec)	String	Returns the parent directory path of the given path. Returns an empty String if there is no parent.
GetSpecialFolder(foldertype)	String	Returns the path of a special system folder: 0 = Windows directory, 1 = System32, 2 = Temp directory.
GetStandardStream(standardStreamType)	TextStream	Returns a TextStream for stdin (0), stdout (1), or stderr (2). Raises error 5 for any other stream type.
GetTempName()	String	Returns a unique temporary file name in `radXXXXXX.axon.tmp` format. Does not create the file on disk.
MoveFile(source, destination)	Empty	Moves a file from source to destination.
MoveFolder(source, destination)	Empty	Moves a folder from source to destination.
OpenTextFile(filename [, iomode [, create]])	TextStream	Opens a text file and returns a TextStream. iomode: 1 = ForReading, 2 = ForWriting, 8 = ForAppending.

601.3 Remarks

- Use `Set` when assigning the return value of any method that returns a File, Folder, TextStream, or Drive object.
- Mutating methods (Delete, Move, Copy) silently ignore operations when the source or destination path cannot be resolved, unless a VBScript error is explicitly raised by the runtime.
- `On Error Resume Next` is the standard guard for file operations that may fail at runtime.

602 Properties

602.1 Overview

This page lists properties exposed by the Scripting.FileSystemObject object family. Properties are organized by the sub-object that exposes them.

602.2 FileSystemObject Properties

Property	Access	Type	Description
Drives	Read	DrivesCollection	Returns a DrivesCollection object containing one Drive entry per available drive on the server.

602.3 File Properties

A File object is returned by `GetFile` or from the `Files` collection of a Folder.

Property	Access	Type	Description
Attributes	Read	Integer	Bitmask of file attributes: 1 = ReadOnly, 2 = Hidden, 32 = Archive.
DateCreated	Read	Date	The date and time the file was created.
DateLastAccessed	Read	Date	The date and time the file was last accessed.

Property	Access	Type	Description
DateLastModified	Read	Date	The date and time the file was last modified.
Drive	Read	Drive	The Drive object for the drive on which the file resides.
Name	Read/Write	String	The file name including its extension. Assigning a new value renames the file within its parent folder.
ParentFolder	Read	Folder	The Folder object representing the parent directory of this file.
Path	Read	String	The full absolute path to the file.
ShortName	Read	String	The short (8.3) file name. On non-Windows hosts this is the same as Name.
ShortPath	Read	String	The full short-format path. On non-Windows hosts this is the same as Path.
Size	Read	Integer	The size of the file in bytes.
Type	Read	String	A description of the file type derived from its extension (for example, “TXT File”). Returns “File” when there is no extension.

602.4Folder Properties

A Folder object is returned by `GetFolder`, `CreateFolder`, or from the `SubFolders` collection of another Folder.

Property	Access	Type	Description
Attributes	Read	Integer	Bitmask of folder attributes: 2 = Hidden, 16 = Directory.
DateCreated	Read	Date	The date and time the folder was created.
DateLastAccessed	Read	Date	The date and time the folder was last accessed.
DateLastModified	Read	Date	The date and time the folder was last modified.
Drive	Read	Drive	The Drive object for the drive on which the folder resides.
Files	Read	FilesCollection	A collection of File objects for all files directly inside this folder.
IsRootFolder	Read	Boolean	Returns True if this folder is the root of its drive; False otherwise.
Name	Read/Write	String	The folder name. Assigning a new value renames the folder within its parent directory.
ParentFolder	Read	Folder	The Folder object representing the parent directory of this folder.
Path	Read	String	The full absolute

Property	Access	Type	Description
			path to the folder.
ShortName	Read	String	The short (8.3) folder name. On non-Windows hosts this is the same as Name.
ShortPath	Read	String	The full short-format path. On non-Windows hosts this is the same as Path.
Size	Read	Integer	The total size of the folder in bytes, including all files and sub-folders.
SubFolders	Read	SubFoldersCollection	A collection of Folder objects for all immediate sub-folders of this folder.
Type	Read	String	Always returns “Folder” for Folder objects.

602.5 TextStream Properties

A TextStream object is returned by `OpenTextFile`, `CreateTextFile`, or `File.OpenAsTextStream`.

Property	Access	Type	Description
AtEndOfStream	Read	Boolean	Returns True when the read position has reached the end of the stream; False otherwise.
Column	Read	Integer	The 1-based column position of the current read/write cursor within the current

Property	Access	Type	Description
			line.
Line	Read	Integer	The 1-based line number of the current read/write cursor position.

602.6Drive Properties

A Drive object is returned by `GetDrive` or by the `Drive` property of a File or Folder.

Property	Access	Type	Description
AvailableSpace	Read	Integer	Bytes available on the drive for the current user.
DriveLetter	Read	String	The single-letter drive identifier (for example, “C”).
DriveType	Read	Integer	Always returns 2 (Fixed) in the current implementation.
FileSystem	Read	String	The file system type string. Returns “NTFS” on Windows; “UnixFS” on other platforms.
FreeSpace	Read	Integer	Total free bytes on the drive.
IsReady	Read	Boolean	Returns True if the drive is accessible; False otherwise.
Path	Read	String	The drive path (for example, “C:” on Windows or “/” on Unix).
RootFolder	Read	Folder	The Folder object representing the root directory of this drive.

Property	Access	Type	Description
SerialNumber	Read	String	The volume serial number string of the drive.
ShareName	Read	String	The network share name for the drive, if applicable. Returns an empty String for local drives.
TotalSize	Read	Integer	Total capacity of the drive in bytes.
VolumeName	Read	String	The volume label of the drive.

602.7 Remarks

- Read-only properties do not accept assignment. Attempting to assign a value to a read-only property is silently ignored by the FSO runtime.
- The `Name` property on File and Folder objects is writable and triggers an OS-level rename when assigned.
- TextStream properties (`AtEndOfStream`, `Line`, `Column`) reflect the internal stream state and are updated automatically by read and write operations.

603 BuildPath Method

603.1 Overview

Concatenates a base path and a file or folder name using the OS path separator and returns the resulting path string.

603.2 Syntax

```
result = fso.BuildPath(path, name)
```

603.3 Parameters

Parameter	Type	Required	Description
path	String	Yes	The base path to which name is appended.
name	String	Yes	The file or folder name to append to path.

603.4 Return Value

Returns a **String** containing the concatenated path. The method uses the OS path separator (\ on Windows, / on Unix). If fewer than two arguments are supplied, returns an empty String.

603.5 How It Works

The method calls the Go `filepath.Join` function on the two arguments. No path validation or resolution is performed; the inputs are joined as-is. The result is not checked for existence on disk.

603.6 Remarks

- `BuildPath` does not create any directory or file on disk.
- The method does not resolve relative paths or validate the result against the web root.
- Use `Server.MapPath` when you need an absolute server path, then pass the result to `BuildPath` if you need to append a sub-path.

603.7 Code Example

```
<%
Option Explicit
Dim fso, fullPath
Set fso = Server.CreateObject("Scripting.FileSystemObject")

fullPath = fso.BuildPath("C:\\data", "report.txt")
Response.Write fullPath
' Output: C:\data\report.txt

Set fso = Nothing
%>
```

604 CopyFile Method

604.1 Overview

Copies one file from a source path to a destination path.

604.2 Syntax

```
fso.CopyFile source, destination [, overwrite]
```

604.3 Parameters

Parameter	Type	Required	Description
source	String	Yes	The full path to the

Parameter	Type	Required	Description
			file to copy.
destination	String	Yes	The full path or target directory for the copied file.
overwrite	Boolean	No	Set to True (default) to overwrite an existing destination file. Set to False to skip if the destination already exists.

604.4 Return Value

Returns **Empty**. The method does not return a value.

604.5 How It Works

Both `source` and `destination` are resolved against the web root. If either path cannot be resolved, the copy operation is silently skipped. When `overwrite` is True, the destination file is replaced if it already exists. Parent directories of the destination are not created automatically.

604.6 Remarks

- The `overwrite` parameter defaults to `True`.
- Use `On Error Resume Next` to catch permission or path errors at runtime.
- To copy an entire directory tree, use `CopyFolder` instead.

604.7 Code Example

```
<%
Option Explicit
Dim fso
Set fso = Server.CreateObject("Scripting.FileSystemObject")

On Error Resume Next
fso.CopyFile Server.MapPath("source.txt"), Server.MapPath("backup/source.
txt"), True
If Err.Number <> 0 Then
    Response.Write "Copy failed: " & Err.Description
End If
On Error GoTo 0
```

```
Set fso = Nothing
%>
```

605 CopyFolder Method

605.1 Overview

Recursively copies a folder and all of its contents to a destination path.

605.2 Syntax

```
fso.CopyFolder source, destination [, overwrite]
```

605.3 Parameters

Parameter	Type	Required	Description
source	String	Yes	The full path to the folder to copy.
destination	String	Yes	The full destination path for the copied folder.
overwrite	Boolean	No	Set to True (default) to overwrite existing files in the destination. Set to False to skip existing files.

605.4 Return Value

Returns **Empty**. The method does not return a value.

605.5 How It Works

Both `source` and `destination` are resolved against the web root. The runtime creates any missing parent directories in the destination path before copying files. All files and sub-folders within `source` are copied recursively. If `overwrite` is False, individual files that already exist at the destination are not replaced.

605.6 Remarks

- The `overwrite` parameter defaults to `True`.
- If either path cannot be resolved, the operation is silently skipped.
- Use `On Error Resume Next` to guard against permission or path errors at runtime.

605.7 Code Example

```
<%
Option Explicit
Dim fso
Set fso = Server.CreateObject("Scripting.FileSystemObject")

On Error Resume Next
fso.CopyFolder Server.MapPath("templates"), Server.MapPath("backup/templa
tes"), True
If Err.Number <> 0 Then
    Response.Write "CopyFolder failed: " & Err.Description
End If
On Error GoTo 0

Set fso = Nothing
%>
```

606 CreateFolder Method

606.1 Overview

Creates the specified folder on disk and returns a Folder object for the newly created directory.

606.2 Syntax

```
Set folder = fso.CreateFolder(folderspec)
```

606.3 Parameters

Parameter	Type	Required	Description
folderspec	String	Yes	The path of the folder to create.

606.4 Return Value

Returns a **Folder** object representing the newly created directory. Returns **Empty** if the path cannot be resolved or if `os.MkdirAll` fails.

606.5 How It Works

The path is resolved against the web root. The runtime calls `os.MkdirAll`, which creates all missing intermediate directories in the hierarchy. If the folder already exists, the call succeeds and returns a Folder object for that existing directory. Use `Set` to capture the returned Folder object.

606.6 Remarks

- All intermediate parent directories are created automatically.

- If the resolved path is empty or cannot be determined, the method returns Empty without raising an error.
- Use `FolderExists` first if you need to distinguish between creating a new folder and receiving an existing one.

606.7 Code Example

```
<%
Option Explicit
Dim fso, fldr
Set fso = Server.CreateObject("Scripting.FileSystemObject")

If Not fso.FolderExists(Server.MapPath("uploads/2025")) Then
    Set fldr = fso.CreateFolder(Server.MapPath("uploads/2025"))
    Response.Write "Created: " & fldr.Path
    Set fldr = Nothing
End If

Set fso = Nothing
%>
```

607 CreateTextFile Method

607.1 Overview

Creates a new text file on disk and returns a TextStream object opened for writing.

607.2 Syntax

```
Set ts = fso.CreateTextFile(filename [, overwrite])
```

607.3 Parameters

Parameter	Type	Required	Description
filename	String	Yes	The full path of the file to create.
overwrite	Boolean	No	Set to True (default) to truncate and overwrite an existing file. Set to False to fail if the file already exists.

607.4 Return Value

Returns a **TextStream** object opened for writing (mode 2). Returns **Empty** if the path cannot be resolved, if `overwrite` is False and the file already exists, or if the OS rejects the file creation.

607.5 How It Works

The path is resolved against the web root. The runtime opens the file with `O_CREATE | O_WRONLY`. When `overwrite` is True, `O_TRUNC` is added to clear any existing content. When `overwrite` is False, `O_EXCL` is added so the open call fails if the file exists. The resulting TextStream starts at position line 1, column 1.

607.6 Remarks

- The returned TextStream is write-only. Calling read methods on it returns Empty or empty String.
- Always call `ts.Close` and then `Set ts = Nothing` when you are finished writing.
- If the parent directory does not exist, file creation fails and the method returns Empty.

607.7 Code Example

```
<%
Option Explicit
Dim fso, ts
Set fso = Server.CreateObject("Scripting.FileSystemObject")

Set ts = fso.CreateTextFile(Server.MapPath("output/log.txt"), True)
If Not IsNull(ts) And Not IsEmpty(ts) Then
    ts.WriteLine "Log started: " & Now
    ts.Close
    Set ts = Nothing
End If

Set fso = Nothing
%>
```

608 DeleteFile Method

608.1 Overview

Deletes the specified file from disk.

608.2 Syntax

```
fso.DeleteFile filespec
```

608.3 Parameters

Parameter	Type	Required	Description
filespec	String	Yes	The full path to the file to delete.

608.4 Return Value

Returns **Empty**. The method does not return a value.

608.5 How It Works

The path is resolved against the web root. If the path resolves to an existing file, the runtime removes it. If the target is a directory instead of a file, the runtime raises VBScript error 53 (File not found). Any open TextStream handles rooted at the same path are closed and released before the delete is attempted.

608.6 Error Conditions

Condition	VBScript Error
File does not exist	53 — File not found
Target is a directory	53 — File not found
Access denied by the OS	70 — Permission denied

608.7 Remarks

- Use `FileExists` before calling `DeleteFile` if you want to avoid raising an error when the file may not exist.
- Use `On Error Resume Next` to handle deletion failures gracefully.

608.8 Code Example

```
<%
Option Explicit
Dim fso, filePath
Set fso = Server.CreateObject("Scripting.FileSystemObject")

filePath = Server.MapPath("temp/cache.tmp")
If fso.FileExists(filePath) Then
    On Error Resume Next
    fso.DeleteFile filePath
    If Err.Number <> 0 Then
        Response.Write "Delete failed: " & Err.Description
    End If
    On Error GoTo 0
End If

Set fso = Nothing
%>
```

609 DeleteFolder Method

609.1 Overview

Deletes the specified folder and all of its contents from disk.

609.2 Syntax

```
fso.DeleteFolder folderspec
```

609.3 Parameters

Parameter	Type	Required	Description
folderspec	String	Yes	The full path to the folder to delete.

609.4 Return Value

Returns **Empty**. The method does not return a value.

609.5 How It Works

The path is resolved against the web root. The runtime first releases any open FSO handles (TextStream, File, or Folder objects) that point to paths within the target directory. It then sets file permissions to writable on all contained entries and removes the directory tree recursively. On Windows, the delete is retried up to five times with a short delay to absorb transient sharing locks.

609.6 Error Conditions

Condition	VBScript Error
Folder does not exist	76 — Path not found
Path resolves to a file	76 — Path not found
Access denied by the OS	70 — Permission denied

609.7 Remarks

- All files and sub-folders inside the target are deleted. This operation cannot be undone.
- Use `FolderExists` before calling `DeleteFolder` to avoid errors when the folder may not exist.
- Use `On Error Resume Next` to handle deletion failures gracefully.

609.8 Code Example

```
<%
Option Explicit
Dim fso, folderPath
Set fso = Server.CreateObject("Scripting.FileSystemObject")
```

```
folderPath = Server.MapPath("temp/session_cache")
If fso.FolderExists(folderPath) Then
    On Error Resume Next
    fso.DeleteFolder folderPath
    If Err.Number <> 0 Then
        Response.Write "Delete failed: " & Err.Description
    End If
    On Error GoTo 0
End If

Set fso = Nothing
%>
```

610 FileExists Method

610.1 Overview

Determines whether a file exists at the specified path.

610.2 Syntax

```
result = fso.FileExists(filespec)
```

610.3 Parameters

Parameter	Type	Required	Description
filespec	String	Yes	The path to the file to check.

610.4 Return Value

Returns **True** if a file exists at the resolved path and the entry is not a directory. Returns **False** if the file does not exist, if the path resolves to a directory, if the path cannot be resolved, or if no argument is supplied.

610.5 How It Works

The path is resolved against the web root. The runtime calls `os.Stat` on the resolved path and returns True only when the call succeeds and the result is not a directory entry. No error is raised; the method always returns a Boolean.

610.6 Remarks

- `FileExists` does not distinguish between a missing file and a permission error. Both cases return False.
- To check for a directory, use `FolderExists` instead.

610.7 Code Example

```
<%
Option Explicit
Dim fso, configPath
Set fso = Server.CreateObject("Scripting.FileSystemObject")

configPath = Server.MapPath("config/settings.ini")
If fso.FileExists(configPath) Then
    Response.Write "Config file found."
Else
    Response.Write "Config file not found."
End If

Set fso = Nothing
%>
```

611 FolderExists Method

611.1 Overview

Determines whether a folder exists at the specified path.

611.2 Syntax

```
result = fso.FolderExists(folderspec)
```

611.3 Parameters

Parameter	Type	Required	Description
folderspec	String	Yes	The path to the folder to check.

611.4 Return Value

Returns **True** if a directory exists at the resolved path. Returns **False** if the directory does not exist, if the path resolves to a file, if the path cannot be resolved, or if no argument is supplied.

611.5 How It Works

The path is resolved against the web root. The runtime calls `os.Stat` on the resolved path and returns True only when the call succeeds and the result is a directory entry. No error is raised; the method always returns a Boolean.

611.6 Remarks

- `FolderExists` does not distinguish between a missing folder and a permission error. Both cases return False.
- To check for a file, use `FileExists` instead.

611.7 Code Example

```
<%
Option Explicit
Dim fso, uploadDir
Set fso = Server.CreateObject("Scripting.FileSystemObject")

uploadDir = Server.MapPath("uploads")
If Not fso.FolderExists(uploadDir) Then
    fso.CreateFolder uploadDir
    Response.Write "Upload folder created."
Else
    Response.Write "Upload folder already exists."
End If

Set fso = Nothing
%>
```

612 GetFile Method

612.1 Overview

Returns a File object for an existing file at the specified path.

612.2 Syntax

```
Set f = fso.GetFile(filespec)
```

612.3 Parameters

Parameter	Type	Required	Description
filespec	String	Yes	The full path to the file.

612.4 Return Value

Returns a **File** object representing the file at the resolved path. Use `Set` for assignment. Raises a VBScript error if the file does not exist or cannot be accessed.

612.5 Error Conditions

Condition	VBScript Error
File does not exist	53 — File not found
Path resolves to a directory	53 — File not found
Access denied by the OS	70 — Permission denied

612.6 How It Works

The path is resolved against the web root. The runtime calls `os.Stat` to confirm that the path exists and is a file. On success, it stores a native `fsoKindFile` object internally and returns the handle. The File object exposes properties such as `Name`, `Size`, `Path`, `DateLastModified`, and methods including `Copy`, `Move`, `Delete`, and `OpenAsTextStream`.

612.7 Remarks

- Always use `Set` to capture the returned File object.
- Call `Set f = Nothing` when the File object is no longer needed.
- To check existence before calling `GetFile`, use `FileExists`.

612.8 Code Example

```
<%
Option Explicit
Dim fso, f
Set fso = Server.CreateObject("Scripting.FileSystemObject")

Set f = fso.GetFile(Server.MapPath("data/report.csv"))
Response.Write f.Name & " — " & f.Size & " bytes" & "<br>"
Response.Write "Modified: " & f.DateLastModified
Set f = Nothing

Set fso = Nothing
%>
```

613 GetFolder Method

613.1 Overview

Returns a Folder object for an existing folder at the specified path.

613.2 Syntax

```
Set fldr = fso.GetFolder(folderspec)
```

613.3 Parameters

Parameter	Type	Required	Description
folderspec	String	Yes	The full path to the folder.

613.4 Return Value

Returns a **Folder** object representing the directory at the resolved path. Use `Set` for assignment. Raises a VBScript error if the folder does not exist or cannot be accessed.

613.5 Error Conditions

Condition	VBScript Error
Folder does not exist	76 — Path not found
Path resolves to a file	76 — Path not found
Access denied by the OS	70 — Permission denied

613.6 How It Works

The path is resolved against the web root. The runtime calls `os.Stat` to confirm that the path exists and is a directory. On success, it stores a native `fsoKindFolder` object internally and returns the handle. The Folder object exposes properties such as `Name`, `Path`, `Files`, `SubFolders`, `Size`, `DateLastModified`, and methods including `Copy`, `Move`, `Delete`, and `CreateTextFile`.

613.7 Remarks

- Always use `Set` to capture the returned Folder object.
- Call `Set fldr = Nothing` when the Folder object is no longer needed.
- To check existence before calling `GetFolder`, use `FolderExists`.

613.8 Code Example

```
<%
Option Explicit
Dim fso, fldr
Set fso = Server.CreateObject("Scripting.FileSystemObject")

Set fldr = fso.GetFolder(Server.MapPath("uploads"))
Response.Write fldr.Name & " — " & fldr.Size & " bytes total"
Set fldr = Nothing

Set fso = Nothing
%>
```

614 GetTempName Method

614.1 Overview

Generates and returns a unique temporary file name. The file is not created on disk.

614.2 Syntax

```
result = fso.GetTempName()
```

614.3 Parameters

None.

614.4 Return Value

Returns a **String** containing a unique temporary file name in the format `radXXXXXXXXXX.axon.tmp`, where `X` characters are hexadecimal digits derived from the current time in nanoseconds.

614.5 How It Works

The method formats the current `time.Now().UnixNano()` value as uppercase hexadecimal and embeds it into the `rad%X.axon.tmp` pattern. The result is a name-only string, not a full path. No file is created, and no disk access is performed.

614.6 Remarks

- The returned name is not a full path. Combine it with `BuildPath` and a directory path (such as the system temp directory from `GetSpecialFolder(2)`) before creating the file.
- Because the name is based on the current nanosecond timestamp, normal web request sequencing makes collisions extremely unlikely, but not impossible under heavy concurrent load. Verify file non-existence with `FileExists` before use if strict uniqueness is required.

614.7 Code Example

```
<%
Option Explicit
Dim fso, tempName, tempPath, ts
Set fso = Server.CreateObject("Scripting.FileSystemObject")

tempName = fso.GetTempName()
tempPath = fso.BuildPath(fso.GetSpecialFolder(2), tempName)

Set ts = fso.CreateTextFile(tempPath, False)
ts.Write "temporary data"
ts.Close
Set ts = Nothing

Response.Write "Temp file: " & tempPath

fso.DeleteFile tempPath
Set fso = Nothing
%>
```

615 MoveFile Method

615.1 Overview

Moves a file from a source path to a destination path.

615.2 Syntax

```
fso.MoveFile source, destination
```

615.3 Parameters

Parameter	Type	Required	Description
source	String	Yes	The full path of the file to move.
destination	String	Yes	The full destination path, including the file name.

615.4 Return Value

Returns **Empty**. The method does not return a value.

615.5 How It Works

Both paths are resolved against the web root. The runtime first attempts `os.Rename` for an atomic move. If the rename fails (for example, across different volumes), the runtime falls back to a copy-then-delete sequence. If the destination already exists, it is replaced. Parent directories of the destination are created automatically during the fallback path.

615.6 Remarks

- If either path cannot be resolved, the operation is silently skipped.
- To move an entire directory, use `MoveFolder` instead.
- Use `On Error Resume Next` to handle errors gracefully at runtime.

615.7 Code Example

```
<%
Option Explicit
Dim fso
Set fso = Server.CreateObject("Scripting.FileSystemObject")

On Error Resume Next
fso.MoveFile Server.MapPath("inbox/order.xml"), Server.MapPath("processed
/order.xml")
If Err.Number <> 0 Then
    Response.Write "Move failed: " & Err.Description
End If
On Error GoTo 0

Set fso = Nothing
%>
```

616 MoveFolder Method

616.1 Overview

Moves a folder and all of its contents from a source path to a destination path.

616.2 Syntax

```
fso.MoveFolder source, destination
```

616.3 Parameters

Parameter	Type	Required	Description
source	String	Yes	The full path of the folder to move.
destination	String	Yes	The full destination path for the moved folder.

616.4 Return Value

Returns **Empty**. The method does not return a value.

616.5 How It Works

Both paths are resolved against the web root. The runtime first attempts `os.Rename` for an atomic move. If the rename fails (for example, across different volumes), the runtime falls back to a recursive copy-then-delete sequence. Any existing destination at that path is replaced. Parent directories of the destination are created automatically during the fallback path.

616.6 Remarks

- If either path cannot be resolved, the operation is silently skipped.
- To move only a single file, use `MoveFile` instead.
- Use `On Error Resume Next` to handle errors gracefully at runtime.

616.7 Code Example

```
<%
Option Explicit
Dim fso
Set fso = Server.CreateObject("Scripting.FileSystemObject")

On Error Resume Next
fso.MoveFolder Server.MapPath("drafts/2024"), Server.MapPath("archive/202
4")
If Err.Number <> 0 Then
```

```
    Response.Write "MoveFolder failed: " & Err.Description
End If
On Error GoTo 0

Set fso = Nothing
%>
```

617 OpenTextFile Method

617.1 Overview

Opens a text file for reading, writing, or appending and returns a TextStream object.

617.2 Syntax

```
Set ts = fso.OpenTextFile(filename [, iomode [, create]])
```

617.3 Parameters

Parameter	Type	Required	Description
filename	String	Yes	The full path to the text file to open.
iomode	Integer	No	The file access mode. 1 = ForReading (default), 2 = ForWriting, 8 = ForAppending.
create	Boolean	No	When True and iomode is 1 (ForReading), creates the file if it does not exist. Ignored for write and append modes, which always create the file. Defaults to False.

617.4 Return Value

Returns a **TextStream** object for the opened file. Returns **Empty** if the path cannot be resolved or if the OS rejects the open call (for example, the file does not exist and `create` is False for read mode).

617.5 How It Works

The path is resolved against the web root. The runtime translates `iomode` into OS open flags:

- **1 (ForReading):** Opens with `O_RDONLY`. If `create` is True, `O_CREATE` is also set.
- **2 (ForWriting):** Opens with `O_WRONLY | O_CREATE | O_TRUNC`, replacing any existing content.
- **8 (ForAppending):** Opens with `O_WRONLY | O_CREATE | O_APPEND`, adding content at the end.

For write and append modes, the runtime creates any missing parent directories automatically.

617.6 Remarks

- A TextStream opened in ForReading mode (1) rejects all write operations (`Write`, `WriteLine`, `WriteBlankLines`). Those calls return Empty silently.
- Always call `ts.Close` and `Set ts = Nothing` after use to release the file handle.
- Use `FileExists` before opening in read mode when you need to handle missing files without an error.

617.7 Code Example

```
<%
Option Explicit
Dim fso, ts, line
Set fso = Server.CreateObject("Scripting.FileSystemObject")

' Append a log entry
Set ts = fso.OpenTextFile(Server.MapPath("logs/access.log"), 8, True)
ts.WriteLine Now & " - page loaded"
ts.Close
Set ts = Nothing

' Read the entire log
Set ts = fso.OpenTextFile(Server.MapPath("logs/access.log"), 1)
Do While Not ts.AtEndOfStream
    line = ts.ReadLine
    Response.Write Server.HtmlEncode(line) & "<br>"
Loop
ts.Close
Set ts = Nothing

Set fso = Nothing
%>
```

618 Use VBScript.RegExp in AxonASP

618.1 Overview

VBScript.RegExp provides regular expression matching, searching, and replacement for Classic ASP scripts. Use it to validate input, extract substrings, and perform pattern-based text transformations.

The object supports the full regular expression syntax accepted by Go's `regexp` package, which follows RE2 semantics. This is compatible with the majority of VBScript regex patterns, with the exception of lookahead and lookbehind assertions and backreferences, which are not supported by RE2.

618.2 Prerequisites

- Set `Pattern` before calling any method. Calling `Execute`, `Test`, or `Replace` without a pattern returns an empty Matches collection, `False`, or the original input string respectively.
- Setting `IgnoreCase` or `MultiLine` after `Pattern` recompiles the internal expression automatically.

618.3 Syntax

```
Set re = Server.CreateObject("VBScript.RegExp")
```

618.4 Return Value

Returns a native RegExp object handle. Use `Set` for assignment. Call `Set re = Nothing` when the object is no longer needed.

618.5 How It Works

The AxonASP runtime stores each RegExp instance in an internal map keyed by a dynamic numeric ID. Setting the `Pattern`, `IgnoreCase`, or `MultiLine` property triggers an immediate recompile of the underlying Go `*regexp.Regexp`. If the pattern is syntactically invalid, the runtime raises VBScript error 5017 (Regular expression syntax error) and clears the compiled expression.

The `Execute` method returns a **MatchesCollection** object. Iterating it yields individual **Match** objects, each exposing `Value`, `FirstIndex`, `Length`, and a `SubMatches` collection.

Replacement token syntax supported by `Replace`:

Token	Replaced with
`$&` or `$0`	The entire matched text

Token	Replaced with
$1, $2, …	Captured group 1, 2, …
$`	Text before the match
$'	Text after the match

618.6 Remarks

- All method and property names are case-insensitive.
- When `Global` is False (default), `Execute` returns at most one match and `Replace` replaces only the first occurrence.
- When `Global` is True, `Execute` returns all non-overlapping matches and `Replace` replaces all occurrences.
- `FirstIndex` on a Match object is 0-based, consistent with classic VBScript behavior.
- Lookahead, lookbehind, and backreferences (`\1`, `\2`) are not supported.

618.7 Code Example

```
<%
Option Explicit
Dim re, matches, m
Set re = Server.CreateObject("VBScript.RegExp")
re.Pattern = "(\\d+)"
re.Global = True

Set matches = re.Execute("Order 123 and invoice 456")
Dim i
For i = 0 To matches.Count - 1
    Set m = matches.Item(i)
    Response.Write "Found: " & m.Value & " at index " & m.FirstIndex & "<
br>"
    Set m = Nothing
Next
Set matches = Nothing
Set re = Nothing
%>
```

619 Methods

619.1 Overview

This page lists all methods exposed by the VBScript.RegExp object.

619.2 RegExp Methods

Method	Returns	Description

Method	Returns	Description
Execute(string)	MatchesCollection	Applies the compiled pattern to the input string and returns a MatchesCollection containing all matches found. When `Global` is False, at most one match is returned.
Replace(string, replacement)	String	Replaces matches of the pattern within `string` with the `replacement` text and returns the resulting String. When `Global` is False, only the first match is replaced.
Test(string)	Boolean	Returns True if the pattern matches anywhere in the input string; returns False if there is no match or the pattern has not been compiled.

619.3 MatchesCollection Methods

A MatchesCollection object is returned by `Execute`. It is read-only.

Method	Returns	Description
Item(index)	Match	Returns the Match object at the specified 0-based index. Returns Empty if the index is out of range.
Count()	Integer	Returns the total number of Match objects in the collection.

619.4 Match Methods

A Match object is obtained from a MatchesCollection via `Item`.

Method	Returns	Description
SubMatches([index])	SubMatches or String	With no argument, returns

Method	Returns	Description
		the SubMatches collection for this match. With an integer argument, returns the String value of the captured group at that 0-based index. Returns Empty if the index is out of range.

619.5 SubMatches Methods

Method	Returns	Description
Item(index)	SubMatchValue	Returns the SubMatchValue object for the captured group at the specified 0-based index. Returns Empty if the index is out of range.
Count()	Integer	Returns the number of captured groups in this SubMatches collection.

619.6 Remarks

- A MatchesCollection, Match, or SubMatches object is a native handle. Use `Set` for assignment.
- The `Item` method is the default dispatch member; `matches(0)` and `matches.Item(0)` are equivalent.

620 Properties

620.1 Overview

This page lists all properties exposed by the VBScript.RegExp object and its sub-objects.

620.2 RegExp Properties

Property	Access	Type	Description
Global	Read/Write	Boolean	When True, Execute finds all non-overlapping matches and

Property	Access	Type	Description
			`Replace` replaces all occurrences. When False (default), only the first match is processed. Changing this property does not recompile the pattern.
IgnoreCase	Read/Write	Boolean	When True, the compiled pattern ignores letter case during matching. When False (default), matching is case-sensitive. Changing this property recompiles the pattern immediately.
MultiLine	Read/Write	Boolean	When True, the ^ and $ anchors match at the start and end of each line within the input string. When False (default), they match only at the start and end of the entire string. Changing this property recompiles the pattern.
Pattern	Read/Write	String	The regular expression pattern string. Setting this property compiles

Property	Access	Type	Description
			the expression immediately. If the pattern is syntactically invalid, VBScript error 5017 is raised and the compiled expression is cleared. Returns an empty String when no pattern has been set.

620.3Match Properties

A Match object is obtained from a MatchesCollection via `Item(index)` after calling `Execute`.

Property	Access	Type	Description
FirstIndex	Read	Integer	The 0-based character position in the input string where the match begins.
Length	Read	Integer	The number of characters in the matched text.
SubMatches	Read	SubMatches	The SubMatches collection containing the text captured by each parenthesized group in the pattern.
Value	Read	String	The complete text of the match. This is the default property; reading a Match object directly returns this

Property	Access	Type	Description
			value.

620.4 MatchesCollection Properties

Property	Access	Type	Description
Count	Read	Integer	The total number of Match objects in the collection. Also accessible as `Length`.

620.5 SubMatches Properties

Property	Access	Type	Description
Count	Read	Integer	The number of captured groups in the SubMatches collection.

620.6 SubMatchValue Properties

A SubMatchValue object is returned by `SubMatches.Item(index)`.

Property	Access	Type	Description
Length	Read	Integer	The number of characters in the captured group text.
Value	Read	String	The text captured by the group. This is the default property.

621 Execute Method

621.1 Overview

Applies the compiled regular expression pattern to an input string and returns a MatchesCollection object containing all matches found.

621.2 Syntax

```
Set matches = re.Execute(string)
```

621.3 Parameters

Parameter	Type	Required	Description
string	String	Yes	The text to search for pattern matches.

621.4 Return Value

Returns a **MatchesCollection** object. Use `Set` for assignment. When no matches are found, the returned collection is empty (`Count` equals 0). Never returns Empty or Nothing.

- When `Global` is **False** (default), the collection contains at most one Match object for the first match found.
- When `Global` is **True**, the collection contains one Match object per non-overlapping match, in order of occurrence.

Each Match object in the collection exposes: - **Value** (String) — the matched text - **FirstIndex** (Integer) — the 0-based start position in the input string - **Length** (Integer) — the character count of the matched text - **SubMatches** (SubMatches) — a collection of captured group strings

621.5 How It Works

The runtime calls `FindStringSubmatchIndex` (first-only mode) or `FindAllStringSubmatchIndex` (global mode) on the compiled Go `*regexp.Regexp`. If the pattern has not yet been compiled (because `Pattern` was never set or was set to an empty string), the method recompiles before executing. If the pattern is still absent, an empty MatchesCollection is returned immediately.

Submatches are extracted from the index pairs starting at position 2 in the index slice. If a capturing group did not participate in the match, its SubMatches slot contains an empty String.

621.6 Remarks

- If `Pattern` is empty, the method returns an empty MatchesCollection without raising an error.
- If `Pattern` is syntactically invalid, VBScript error 5017 is raised and an empty MatchesCollection is returned.
- Accessing a MatchesCollection or Match object requires `Set` for assignment.
- `matches.Item(0)` and `matches(0)` are equivalent; `Item` is the default dispatch member.

621.7 Code Example

```
<%
Option Explicit
Dim re, matches, m, i
Set re = Server.CreateObject("VBScript.RegExp")
re.Pattern = "([A-Za-z]+)@([A-Za-z]+\\.com)"
re.Global = True
re.IgnoreCase = True

Set matches = re.Execute("Contact us at info@example.com or sales@company.
com")
For i = 0 To matches.Count - 1
    Set m = matches.Item(i)
    Response.Write "Email: " & m.Value & "<br>"
    Response.Write "User: " & m.SubMatches(0) & "<br>"
    Response.Write "Domain: " & m.SubMatches(1) & "<br>"
    Set m = Nothing
Next
Set matches = Nothing
Set re = Nothing
%>
```

622 Replace Method

622.1 Overview

Replaces pattern matches within an input string with a replacement string and returns the resulting text.

622.2 Syntax

```
result = re.Replace(string, replacement)
```

622.3 Parameters

Parameter	Type	Required	Description
string	String	Yes	The source text in which to perform replacements.
replacement	String	Yes	The replacement text. Supports substitution tokens (see Remarks).

622.4 Return Value

Returns a **String** containing the source text after replacements have been applied.

- When `Global` is **False** (default), only the first match is replaced.
- When `Global` is **True**, all non-overlapping matches are replaced.
- If the pattern does not match anywhere in `string`, the original `string` is returned unchanged.
- If fewer than two arguments are supplied, the method returns the first argument as-is, or an empty String if no arguments are given.

622.5 How It Works

The runtime locates match positions using `FindStringSubmatchIndex` or `FindAllStringSubmatchIndex` and rebuilds the string by inserting the processed replacement between each unmatched segment. Before insertion, the replacement string is scanned for substitution tokens and each token is expanded:

Token	Expanded value
`$&` or `$0`	The entire matched text.
`$1`, `$2`, …	The text of captured group 1, 2, … If the group did not participate in the match, an empty String is substituted.
`$``	The text in `string` before the match.
`$'`	The text in `string` after the match.

622.6 Remarks

- If `Pattern` is empty, the original `string` is returned without modification.
- If `Pattern` is syntactically invalid, VBScript error 5017 is raised and the original `string` is returned.
- Replacement tokens are processed case-sensitively. `$1` is never the same as `$01`.
- To include a literal dollar sign in the output, use `$$` is not supported; use a different delimiter or post-process the result.

622.7 Code Example

```
<%
Option Explicit
Dim re, result
Set re = Server.CreateObject("VBScript.RegExp")
re.Pattern = "(\\d{4})-(\\d{2})-(\\d{2})"
re.Global = False

' Reformat ISO date to DD/MM/YYYY
result = re.Replace("Event date: 2025-12-31", "$3/$2/$1")
Response.Write result
' Output: Event date: 31/12/2025
```

```
Set re = Nothing
%>
```

623 Test Method

623.1 Overview

Tests whether the compiled regular expression pattern matches anywhere within an input string.

623.2 Syntax

```
result = re.Test(string)
```

623.3 Parameters

Parameter	Type	Required	Description
string	String	Yes	The text to test against the pattern.

623.4 Return Value

Returns **True** if the pattern matches at least one position in the input string. Returns **False** if there is no match, if no argument is supplied, if `Pattern` is empty, or if the pattern failed to compile.

623.5 How It Works

The runtime calls `MatchString` on the compiled Go `*regexp.Regexp`. If the pattern has not yet been compiled, the runtime attempts to compile it first. If compilation fails, `False` is returned and VBScript error 5017 is raised.

The `Global` property has no effect on `Test`. The method only determines presence of a match, not count.

623.6 Remarks

- `Test` is the fastest way to check for a match when the matched text is not needed. It performs no allocations beyond the initial compile.
- Unlike `Execute`, `Test` never returns an object. It always returns a Boolean.
- `IgnoreCase` and `MultiLine` apply to the pattern used by `Test` exactly as they do for `Execute`.

623.7 Code Example

```
<%
Option Explicit
Dim re
Set re = Server.CreateObject("VBScript.RegExp")
```

```
re.Pattern = "^[a-zA-Z0-9._%+\\-]+@[a-zA-Z0-9.\\-]+\\.[a-zA-Z]{2,}$"
re.IgnoreCase = True

Dim email
email = Request.Form("email")
If re.Test(email) Then
    Response.Write "Valid email address."
Else
    Response.Write "Invalid email address."
End If

Set re = Nothing
%>
```

624 Global Property

624.1 Overview

Gets or sets whether the regular expression operates in global mode, applying to all matches in the input string rather than only the first.

624.2 Syntax

```
re.Global = True
value = re.Global
```

624.3 Return Value

Returns a **Boolean** indicating the current global mode setting. Returns **False** by default when the object is first created.

624.4 How It Works

Setting `Global` to True or False updates the internal flag only. **The pattern is not recompiled** when `Global` changes. The flag is read at the moment `Execute` or `Replace` is called:

- `Execute` with `Global = True` calls `FindAllStringSubmatchIndex` to collect all non-overlapping matches.
- `Execute` with `Global = False` calls `FindStringSubmatchIndex` and returns at most one match.
- `Replace` with `Global = True` replaces all matches; `Global = False` replaces only the first.
- `Test` is unaffected by `Global`.

624.5 Remarks

- The default value is **False**.

- Changing `Global` does not trigger pattern recompilation. Only `Pattern`, `IgnoreCase`, and `MultiLine` trigger recompilation.

624.6 Code Example

```
<%
Option Explicit
Dim re, result
Set re = Server.CreateObject("VBScript.RegExp")
re.Pattern = "\\d+"
re.Global = True

result = re.Replace("Room 101, Floor 3, Unit 42", "#")
Response.Write result
' Output: Room #, Floor #, Unit #

Set re = Nothing
%>
```

625 IgnoreCase Property

625.1 Overview

Gets or sets whether the regular expression matching is case-insensitive.

625.2 Syntax

```
re.IgnoreCase = True
value = re.IgnoreCase
```

625.3 Return Value

Returns a **Boolean** indicating the current case-sensitivity setting. Returns **False** by default when the object is first created.

625.4 How It Works

Setting `IgnoreCase` to True prepends the `(?i)` flag to the compiled pattern. Setting it to False removes that flag. After any change to this property, the pattern is recompiled immediately using the current `Pattern` and `MultiLine` values. If `Pattern` is empty, no compilation is triggered.

625.5 Remarks

- The default value is **False** (case-sensitive matching).
- Changing `IgnoreCase` after `Pattern` has been set recompiles the expression in-place.
- `IgnoreCase` affects `Execute`, `Test`, and `Replace` equally.

625.6 Code Example

```
<%
Option Explicit
Dim re
Set re = Server.CreateObject("VBScript.RegExp")
re.Pattern = "hello"
re.IgnoreCase = True

If re.Test("Say HELLO world") Then
    Response.Write "Match found (case-insensitive)."
End If

Set re = Nothing
%>
```

626 MultiLine Property

626.1 Overview

Gets or sets whether the ^ and $ anchors match at the start and end of each line within the input string, rather than only at the very start and end of the entire string.

626.2 Syntax

```
re.MultiLine = True
value = re.MultiLine
```

626.3 Return Value

Returns a **Boolean** indicating the current multi-line mode setting. Returns **False** by default when the object is first created.

626.4 How It Works

Setting `MultiLine` to True prepends the `(?m)` flag to the compiled pattern. Setting it to False removes that flag. After any change to this property, the pattern is recompiled immediately using the current `Pattern` and `IgnoreCase` values.

With `MultiLine = True`: - ^ matches at position 0 and immediately after any `\n` character. - $ matches immediately before any `\n` character and at the end of the string.

626.5 Remarks

- The default value is **False**.
- Changing `MultiLine` after `Pattern` has been set triggers immediate recompilation.
- `MultiLine` applies equally to `Execute`, `Test`, and `Replace`.

- Dot (.) does **not** match newlines regardless of this setting. Use `[\\s\\S]` to match any character including newlines.

626.6 Code Example

```
<%
Option Explicit
Dim re, matches
Set re = Server.CreateObject("VBScript.RegExp")
re.Pattern = "^Item"
re.Global = True
re.MultiLine = True

Dim text
text = "Item A" & Chr(10) & "Item B" & Chr(10) & "Other C"

Set matches = re.Execute(text)
Response.Write "Lines starting with 'Item': " & matches.Count
' Output: Lines starting with 'Item': 2

Set matches = Nothing
Set re = Nothing
%>
```

627 Pattern Property

627.1 Overview

Gets or sets the regular expression pattern string used for matching, replacement, and testing operations.

627.2 Syntax

```
re.Pattern = "expression"
value = re.Pattern
```

627.3 Return Value

Returns a **String** containing the current pattern. Returns an empty String when no pattern has been assigned.

627.4 How It Works

Assigning a value to `Pattern` immediately compiles the expression against the current `IgnoreCase` and `MultiLine` flags. If the pattern is syntactically invalid, VBScript error 5017 (Regular expression syntax error) is raised and the compiled expression is cleared, causing subsequent `Execute`, `Test`, and `Replace` calls to fail gracefully rather than match unpredictably.

Setting `Pattern` to an empty String clears the compiled expression without raising an error.

627.5 Remarks

- The pattern uses RE2 syntax. Lookahead (`(?=...)`, `(?!...)`), lookbehind (`(?<=...)`, `(?<!...)`), and backreferences (`\\1`) are **not supported** and will raise error 5017 when assigned.
- Character classes, quantifiers, anchors, and named groups (`(?P<name>...)`) are fully supported.
- Changing `IgnoreCase` or `MultiLine` after setting `Pattern` recompiles the expression automatically.

627.6 Code Example

```
<%
Option Explicit
Dim re
Set re = Server.CreateObject("VBScript.RegExp")
re.Pattern = "\\b[A-Z]{2,}\\b"
re.Global = True

Response.Write re.Pattern
' Output: \b[A-Z]{2,}\b

Set re = Nothing
%>
```

628 MSXML2 Library

The MSXML2 library provides HTTP client and XML Document Object Model (DOM) support for Classic ASP pages. It exposes five cooperating objects â€” **ServerXMLHTTP**, **DOMDocument**, **XMLNodeList**, **XMLElement**, and **ParseError** â€” that together allow pages to issue HTTP requests, parse XML, traverse document trees, and evaluate XPath expressions.

628.1 Supported ProgIDs

ProgID	Object
`MSXML2.ServerXMLHTTP`	HTTP client
`MSXML2.DOMDocument`	XML document
`MSXML2.DOMDocument.3.0`	XML document (versioned alias)
`MSXML2.DOMDocument.6.0`	XML document (versioned alias)
`Microsoft.XMLDOM`	XML document (legacy alias)

628.2Prerequisites

No additional installation is required. The library is built into AxonASP.

628.3Object Model

ServerXMLHTTP â€” Sends HTTP requests and exposes the response as text, XML, or raw bytes. Create with `Server.CreateObject("MSXML2.ServerXMLHTTP")`.

DOMDocument â€” Parses an XML string or URL into a traversable document tree. Supports XPath queries via `SelectSingleNode` and `SelectNodes`. Create with one of the DOMDocument ProgIDs.

XMLNodeList â€” An indexed, iterable list of `XMLElement` nodes returned by `GetElementsByTagName`, `SelectNodes`, `ChildNodes`, or `Children`.

XMLElement â€” A single node in the document tree. Exposes the node name, value, attributes, and child collection, and supports XPath traversal.

ParseError â€” Returned by `DOMDocument.ParseError`. Always present; check `ErrorCode = 0` to confirm successful parsing.

628.4Remarks

- All method and property names are case-insensitive.
- `DOMDocument.Async` is accepted for compatibility but synchronous behaviour is always used.
- XPath support covers child/descendant axes, attribute predicates, position predicates, `contains()`, `starts-with()`, `not()`, and, or, namespace prefixes via `SelectionNamespaces`, and the `//` shorthand.

628.5Code Examples

Send an HTTP GET request and read the response:

```
<%
Dim oHTTP
Set oHTTP = Server.CreateObject("MSXML2.ServerXMLHTTP")
oHTTP.Open "GET", "https://example.com/api/data", False
oHTTP.Send
If oHTTP.Status = 200 Then
    Response.Write oHTTP.ResponseText
End If
Set oHTTP = Nothing
%>
```

Load and query an XML document:

```
<%
Dim oXML, oNode
Set oXML = Server.CreateObject("MSXML2.DOMDocument")
If oXML.LoadXML("<root><item id='1'>Alpha</item><item id='2'>Beta</item><
/root>") Then
    Set oNode = oXML.SelectSingleNode("//item[@id='2']")
    If Not IsNull(oNode) Then
        Response.Write oNode.Text
    End If
End If
Set oXML = Nothing
%>
```

629 MSXML2 Methods

629.1 ServerXMLHTTP Methods

Method	Returns	Description
`Open(method, url [,async [,user [,password]]])`	Empty	Configures the HTTP method, URL, and optional async flag and credentials.
`SetRequestHeader(header, value)`	Empty	Adds or replaces an outgoing request header.
`Send([body])`	Empty	Executes the configured HTTP request. Accepts an optional String or byte-array body.
`Abort()`	Empty	Cancels the current request without waiting for a response.
`GetResponseHeader(header)`	String	Returns the value of a single response header (case-insensitive lookup).
`GetAllResponseHeaders()`	String	Returns all response headers as a CRLF-delimited `Header: Value` string.

629.2 DOMDocument Methods

Method	Returns	Description
`LoadXML(xmlString)`	Boolean	Parses an XML string into the document. Returns True on success.

Method	Returns	Description
`Load(url)`	Boolean	Loads XML from a file path or HTTP/HTTPS URL. Returns True on success.
`Save(filename)`	Boolean	Serializes the document and writes it to a file. Returns True on success.
`GetElementsByTagName(tagName)`	XMLNodeList	Returns all descendant elements matching the tag name (case-insensitive).
`CreateElement(tagName)`	XMLElement	Creates a new, unattached element node.
`CreateTextNode(text)`	XMLElement	Creates a new text node with the supplied string value.
`CreateAttribute(name)`	XMLElement	Creates a new, unattached attribute node.
`AppendChild(child)`	XMLElement	Attaches a child element to the document root and returns it.
`SelectSingleNode(xpath)`	XMLElement	Returns the first node matching the XPath expression, or Null.
`SelectNodes(xpath)`	XMLNodeList	Returns all nodes matching the XPath expression.
`GetProperty(name)`	Variant	Returns the value of a named document property.
`SetProperty(name, value)`	Empty	Sets a named document property (e.g., `SelectionNamespaces`).

629.3XMLNodeList Methods

Method	Returns	Description
`Item(index)`	XMLElement	Returns the node at the given zero-based index, or Null if out of range.

Method	Returns	Description
`NextNode()`	XMLElement	Returns the next node in a forward-only iteration, or Null when exhausted.

629.4 XMLElement Methods

Method	Returns	Description
`AppendChild(child)`	XMLElement	Appends a child node to this element and returns the child.
`GetElementsByTagName(tagName)`	XMLNodeList	Returns all descendant elements matching the tag name.
`Item(index)`	XMLElement	Returns the direct child at the given zero-based index.
`SetAttribute(name, value)`	Empty	Creates or replaces an attribute on this element.
`GetAttribute(name)`	String	Returns the value of the named attribute, or an empty String if not found.
`RemoveAttribute(name)`	Empty	Removes the named attribute from this element.
`SelectSingleNode(xpath)`	XMLElement	Returns the first node matching the XPath expression relative to this element, or Null.
`SelectNodes(xpath)`	XMLNodeList	Returns all nodes matching the XPath expression relative to this element.

629.5 Method List by Object

629.5.1 ServerXMLHTTP

- Open
- SetRequestHeader
- Send
- Abort

- GetResponseHeader
- GetAllResponseHeaders

629.5.2 DOMDocument

- GetProperty
- SetProperty
- LoadXML
- Load
- Save
- GetElementsByTagName
- CreateElement
- CreateTextNode
- CreateAttribute
- AppendChild
- SelectSingleNode
- SelectNodes

629.5.3 XMLNodeList

- Item
- NextNode

629.5.4 XMLElement

- AppendChild
- GetElementsByTagName
- Item
- SetAttribute
- GetAttribute
- RemoveAttribute
- SelectSingleNode
- SelectNodes

629.6 Remarks

- Method names are case-insensitive.
- Runtime behavior follows VM compatibility implementations for each object wrapper.

630 MSXML2 Properties

630.1 ServerXMLHTTP Properties

Property	Access	Type	Description
`ResponseText`	Read	String	The response body

Property	Access	Type	Description
			decoded as a text string.
`ResponseXML`	Read	DOMDocument	The response body parsed as an XML document. Returns a String if the response is not valid XML.
`ResponseBody`	Read	Byte Array	The raw response body as a byte array.
`Status`	Read	Integer	The HTTP status code (e.g., 200, 404). Returns 0 on a connection error.
`StatusText`	Read	String	The full HTTP status line (e.g., `200 OK`).
`ReadyState`	Read	Integer	The request lifecycle state: 0 = Uninitialized, 1 = Open, 2 = Sent, 3 = Receiving, 4 = Complete.
`Timeout`	Read/Write	Integer	Request timeout in seconds. Default is 30.

630.2 DOMDocument Properties

Property	Access	Type	Description
`DocumentElement`	Read	XMLElement	The root element of the parsed document, or Null if the document is empty.
`XML`	Read	String	The stored XML source or serialized document tree.
`ParseError`	Read	ParseError	The ParseError

Property	Access	Type	Description
			object from the last load or parse. Check `ErrorCode = 0` for success.
`Async`	Read/Write	Boolean	Accepted for compatibility; the implementation always behaves synchronously. Default is False.
`ServerHTTPRequest`	Read/Write	Boolean	Controls whether HTTP requests use the server HTTP stack.
`ResolveExternals`	Read/Write	Boolean	Controls resolution of external entity references.
`ValidateOnParse`	Read/Write	Boolean	Controls DTD validation during parsing.
`PreserveWhiteSpace`	Read/Write	Boolean	Controls whether insignificant whitespace is preserved in the DOM tree.
`SelectionLanguage`	Read/Write	String	The query language used by `SelectSingleNode` and `SelectNodes`. Default is `XPath`.
`SelectionNamespaces`	Read/Write	String	Namespace prefix bindings for XPath queries. Format: `xmlns:prefix='uri'`.

630.3XMLNodeList Properties

Property	Access	Type	Description
`Length`	Read	Integer	The number of nodes in the list.

Property	Access	Type	Description
Count	Read	Integer	Alias for Length.

630.4 ParseError Properties

Property	Access	Type	Description
ErrorCode	Read	Integer	0 = no error. Negative values or HTTP status codes indicate parse or load failures.
Reason	Read	String	A human-readable description of the parse error.
FilePos	Read	Integer	The byte offset in the source where the error was detected.
Line	Read	Integer	The 1-based line number in the source where the error was detected.
LinePos	Read	Integer	The 1-based column position within the error line.
SrcText	Read	String	The source text that was being parsed when the error occurred.
URL	Read	String	The URL or file path that was being loaded when the error occurred.

630.5 XMLElement Properties

Property	Access	Type	Description
NodeName	Read	String	The element tag name or #text for text nodes.

Property	Access	Type	Description
NodeValue	Read/Write	String	The text value of the node. Settable for text and attribute nodes.
Text	Read/Write	String	The concatenated text content of the element and all its descendants.
XML	Read	String	The serialized XML markup of this node and its subtree.
Attributes	Read	Collection	A collection of attribute name/value pairs.
ChildNodes	Read	XMLNodeList	A list of all direct child nodes.
Children	Read	XMLNodeList	Alias for ChildNodes.
FirstChild	Read	XMLElement	The first direct child node, or Null if there are no children.
LastChild	Read	XMLElement	The last direct child node, or Null if there are no children.
ParentNode	Read	XMLElement	The parent element, or Null if this is the root.
Length	Read	Integer	The number of direct child nodes.

631 ServerXMLHTTP.Open Method

Configures the HTTP method, URL, and optional credentials for a pending request. Must be called before Send.

631.1 Syntax

```
objHTTP.Open method, url [, async [, user [, password]]]
```

631.2 Parameters

Parameter	Type	Required	Description
`method`	String	Yes	The HTTP verb in uppercase (e.g., `"GET"`, `"POST"`, `"PUT"`, `"DELETE"`). The value is automatically converted to uppercase.
`url`	String	Yes	The absolute URL of the resource to request.
`async`	Boolean	No	Accepted for compatibility. The implementation always behaves synchronously regardless of this value.
`user`	String	No	Username for HTTP Basic authentication.
`password`	String	No	Password for HTTP Basic authentication.

631.3 Return Value

Empty. This method does not return a value.

631.4 Remarks

- Calling `Open` resets the object state: `ReadyState` is set to `1` and any previously stored response data is cleared.
- All previously set request headers are preserved between `Open` calls.
- Method names are case-insensitive.

631.5 Code Example

```
<%
Dim oHTTP
Set oHTTP = Server.CreateObject("MSXML2.ServerXMLHTTP")
oHTTP.Open "GET", "https://example.com/api/items", False
oHTTP.SetRequestHeader "Accept", "application/xml"
oHTTP.Send
If oHTTP.Status = 200 Then
    Response.Write oHTTP.ResponseText
End If
Set oHTTP = Nothing
%>
```

632 ServerXMLHTTP.SetRequestHeader Method

Adds or replaces a request header that will be sent with the next `Send` call.

632.1 Syntax

```
objHTTP.SetRequestHeader header, value
```

632.2 Parameters

Parameter	Type	Required	Description
`header`	String	Yes	The name of the HTTP header (e.g., `"Content-Type"`, `"Accept"`).
`value`	String	Yes	The value to assign to the header.

632.3 Return Value

Empty. This method does not return a value.

632.4 Remarks

- If a header with the same name is already set, its value is replaced.
- Headers are stored until `Open` is called on a new request or the object is released.
- Method names are case-insensitive.

632.5 Code Example

```
<%
Dim oHTTP
Set oHTTP = Server.CreateObject("MSXML2.ServerXMLHTTP")
oHTTP.Open "GET", "https://example.com/data.xml", False
```

```
oHTTP.SetRequestHeader "Accept", "application/xml"
oHTTP.SetRequestHeader "Authorization", "Bearer mytoken"
oHTTP.Send
Response.Write oHTTP.ResponseText
Set oHTTP = Nothing
%>
```

633 ServerXMLHTTP.Send Method

Executes the HTTP request configured by `Open`. Blocks until the response is received or the timeout expires.

633.1 Syntax

```
objHTTP.Send [body]
```

633.2 Parameters

Parameter	Type	Required	Description
body	String or Byte Array	No	The request body to send. For `POST`/`PUT` requests, supply the payload here. String values are encoded as UTF-8. Byte arrays are sent as-is.

633.3 Return Value

Empty. This method does not return a value. Read `ResponseText`, `ResponseBody`, or `Status` after `Send` completes.

633.4 Remarks

- If no `Content-Type` header is set and a String body is provided, `application/x-www-form-urlencoded` is used automatically.
- If no `Content-Type` header is set and a byte array body is provided, `application/octet-stream` is used automatically.
- A default `User-Agent` header is added automatically if none is set.
- The request timeout defaults to 30 seconds and can be changed via the `Timeout` property before calling `Send`.
- On completion, `ReadyState` is set to 4.
- Method names are case-insensitive.

633.5 Code Example

```
<%
Dim oHTTP
Set oHTTP = Server.CreateObject("MSXML2.ServerXMLHTTP")
oHTTP.Open "POST", "https://example.com/api/submit", False
oHTTP.SetRequestHeader "Content-Type", "application/json"
oHTTP.Send "{\"key\":\"value\"}"
If oHTTP.Status = 200 Then
    Response.Write oHTTP.ResponseText
Else
    Response.Write "Error: " & oHTTP.Status & " " & oHTTP.StatusText
End If
Set oHTTP = Nothing
%>
```

634 ServerXMLHTTP.Abort Method

Cancels the current HTTP request immediately without waiting for a response.

634.1 Syntax

```
objHTTP.Abort
```

634.2 Parameters

None.

634.3 Return Value

Empty. This method does not return a value.

634.4 Remarks

- Calling `Abort` sets `ReadyState` to 4 without populating any response properties.
- Method names are case-insensitive.

634.5 Code Example

```
<%
Dim oHTTP
Set oHTTP = Server.CreateObject("MSXML2.ServerXMLHTTP")
oHTTP.Open "GET", "https://example.com/large-file", False
On Error Resume Next
oHTTP.Abort
On Error GoTo 0
Set oHTTP = Nothing
%>
```

635 ServerXMLHTTP.GetResponseHeader Method

Returns the value of a single response header by name.

635.1 Syntax

```
value = objHTTP.GetResponseHeader(header)
```

635.2 Parameters

Parameter	Type	Required	Description
header	String	Yes	The name of the response header to retrieve. The lookup is case-insensitive.

635.3 Return Value

String. The value of the specified header. Returns an empty String if the header is not present in the response.

635.4 Remarks

- Call this method only after Send has completed (ReadyState = 4).
- The header name lookup is case-insensitive.
- Method names are case-insensitive.

635.5 Code Example

```
<%
Dim oHTTP, sContentType
Set oHTTP = Server.CreateObject("MSXML2.ServerXMLHTTP")
oHTTP.Open "GET", "https://example.com/data", False
oHTTP.Send
sContentType = oHTTP.GetResponseHeader("Content-Type")
Response.Write "Content-Type: " & sContentType
Set oHTTP = Nothing
%>
```

636 ServerXMLHTTP.GetAllResponseHeaders Method

Returns all response headers as a single CRLF-delimited string.

636.1 Syntax

```
headers = objHTTP.GetAllResponseHeaders()
```

636.2 Parameters

None.

636.3 Return Value

String. All response headers concatenated in `Header: Value\r\n` format. Returns an empty String if no response has been received.

636.4 Remarks

- Call this method only after `Send` has completed (`ReadyState = 4`).
- Each line is of the form `Header-Name: value` followed by a carriage return and line feed (`\r\n`).
- Method names are case-insensitive.

636.5 Code Example

```
<%
Dim oHTTP, sHeaders
Set oHTTP = Server.CreateObject("MSXML2.ServerXMLHTTP")
oHTTP.Open "GET", "https://example.com/", False
oHTTP.Send
sHeaders = oHTTP.GetAllResponseHeaders()
Response.Write "<pre>" & Server.HTMLEncode(sHeaders) & "</pre>"
Set oHTTP = Nothing
%>
```

637 DOMDocument.GetProperty Method

Returns the current value of a named document property.

637.1 Syntax

```
value = objXML.GetProperty(name)
```

637.2 Parameters

Parameter	Type	Required	Description
`name`	String	Yes	The name of the property to retrieve (e.g., `"SelectionLanguage"`, `"SelectionNamespaces"`).

637.3 Return Value

Variant. The current value of the named property. Returns Empty if the property name is not recognized.

637.4 Remarks

- This method is equivalent to reading the corresponding named property directly (e.g., `objXML.SelectionNamespaces`).
- Recognised property names: `SelectionLanguage`, `SelectionNamespaces`, `ResolveExternals`, `ValidateOnParse`, `PreserveWhiteSpace`, `ServerHTTPRequest`, `Async`.
- Method names are case-insensitive.

637.5 Code Example

```
<%
Dim oXML, sLang
Set oXML = Server.CreateObject("MSXML2.DOMDocument")
sLang = oXML.GetProperty("SelectionLanguage")
Response.Write "SelectionLanguage: " & sLang
Set oXML = Nothing
%>
```

638 DOMDocument.SetProperty Method

Sets a named document property that controls parsing and query behaviour.

638.1 Syntax

```
objXML.SetProperty name, value
```

638.2 Parameters

Parameter	Type	Required	Description
`name`	String	Yes	The name of the property to set.
`value`	Variant	Yes	The new value for the property.

638.3 Return Value

Empty. This method does not return a value.

638.4 Remarks

- This method is equivalent to assigning the corresponding named property directly.
- The most commonly used property is `SelectionNamespaces`, which binds namespace prefixes for XPath queries. Format: `xmlns:prefix='uri'`.

- Setting an unknown property name is silently ignored.
- Method names are case-insensitive.

638.5 Code Example

```
<%
Dim oXML, oNode
Set oXML = Server.CreateObject("MSXML2.DOMDocument")
oXML.LoadXML "<ns:root xmlns:ns='urn:test'><ns:item>Value</ns:item></ns:r
oot>"
oXML.SetProperty "SelectionNamespaces", "xmlns:ns='urn:test'"
Set oNode = oXML.SelectSingleNode("//ns:item")
If Not IsNull(oNode) Then
    Response.Write oNode.Text
End If
Set oXML = Nothing
%>
```

639 DOMDocument.LoadXML Method

Parses an XML string and loads it into the document object.

639.1 Syntax

```
bSuccess = objXML.LoadXML(xmlString)
```

639.2 Parameters

Parameter	Type	Required	Description
`xmlString`	String	Yes	A well-formed XML string to parse.

639.3 Return Value

Boolean. Returns True if the XML was parsed successfully. Returns False if the string is empty, Null, or contains invalid XML. Check `ParseError.ErrorCode` for details on failure.

639.4 Remarks

- On a failed parse, `ParseError` is populated with the error code, reason, and the source position where the parse stopped.
- Calling `LoadXML` replaces any previously loaded or built document.
- Method names are case-insensitive.

639.5 Code Example

```
<%
Dim oXML
Set oXML = Server.CreateObject("MSXML2.DOMDocument")
If oXML.LoadXML("<catalog><item>First</item><item>Second</item></catalog>
```

```
") Then
    Dim oRoot
    Set oRoot = oXML.DocumentElement
    Response.Write "Root: " & oRoot.NodeName
Else
    Response.Write "Parse error: " & oXML.ParseError.Reason
End If
Set oXML = Nothing
%>
```

640 DOMDocument.Load Method

Loads an XML document from a file path or HTTP/HTTPS URL.

640.1 Syntax

```
bSuccess = objXML.Load(url)
```

640.2 Parameters

Parameter	Type	Required	Description
`url`	String	Yes	An HTTP or HTTPS URL, or a relative or absolute file path resolved via `Server.MapPath`.

640.3 Return Value

Boolean. Returns True if the document was loaded and parsed successfully. Returns False on a network failure, file-not-found error, or invalid XML. `ParseError.ErrorCode` is set to the HTTP status code on HTTP failures, or `-1` for file/parse failures.

640.4 Remarks

- HTTP and HTTPS requests use a 30-second timeout.
- File paths are resolved using `Server.MapPath` from the currently executing page context.
- Calling `Load` replaces any previously loaded or built document.
- Method names are case-insensitive.

640.5 Code Example

```
<%
Dim oXML
Set oXML = Server.CreateObject("MSXML2.DOMDocument")
If oXML.Load("https://example.com/feed.xml") Then
```

```
    Response.Write "Loaded: " & oXML.DocumentElement.NodeName
Else
    Response.Write "Failed: " & oXML.ParseError.Reason
End If
Set oXML = Nothing
%>
```

641 DOMDocument.Save Method

Serializes the document and writes it to a file at the specified path.

641.1 Syntax

```
bSuccess = objXML.Save(filename)
```

641.2 Parameters

Parameter	Type	Required	Description
`filename`	String	Yes	An absolute or relative file path to write the XML to.

641.3 Return Value

Boolean. Returns True if the file was written successfully. Returns False if the document has no content, the context is unavailable, or the file write fails.

641.4 Remarks

- The file is written with UTF-8 encoding.
- Method names are case-insensitive.

641.5 Code Example

```
<%
Dim oXML
Set oXML = Server.CreateObject("MSXML2.DOMDocument")
oXML.LoadXML "<data><entry>Test</entry></data>"
If oXML.Save(Server.MapPath("output.xml")) Then
    Response.Write "Saved."
Else
    Response.Write "Save failed."
End If
Set oXML = Nothing
%>
```

642 DOMDocument.GetElementsByTagName Method

Returns all descendant elements in the document that match the given tag name.

642.1 Syntax

```
Set oList = objXML.GetElementsByTagName(tagName)
```

642.2 Parameters

Parameter	Type	Required	Description
tagName	String	Yes	The element tag name to search for. The match is case-insensitive.

642.3 Return Value

XMLNodeList. A list of all matching elements, ordered by document order. An empty XMLNodeList is returned if the document has no root or no elements match.

642.4 Remarks

- The search traverses the entire document tree, not just direct children.
- Method names are case-insensitive.

642.5 Code Example

```
<%
Dim oXML, oList, i
Set oXML = Server.CreateObject("MSXML2.DOMDocument")
oXML.LoadXML "<books><book>A</book><book>B</book><book>C</book></books>"
Set oList = oXML.GetElementsByTagName("book")
For i = 0 To oList.Length - 1
    Response.Write oList.Item(i).Text & "<br>"
Next
Set oXML = Nothing
%>
```

643 DOMDocument.CreateElement Method

Creates a new, unattached element node with the given tag name.

643.1 Syntax

```
Set oElement = objXML.CreateElement(tagName)
```

643.2 Parameters

Parameter	Type	Required	Description
tagName	String	Yes	The tag name for the new element.

643.3 Return Value

XMLElement. A new element node that is not yet attached to any document tree. Call `AppendChild` to attach it.

643.4 Remarks

- Method names are case-insensitive.

643.5 Code Example

```
<%
Dim oXML, oRoot, oChild
Set oXML = Server.CreateObject("MSXML2.DOMDocument")
Set oRoot = oXML.CreateElement("root")
Set oChild = oXML.CreateElement("item")
oChild.Text = "Hello"
oRoot.AppendChild oChild
oXML.AppendChild oRoot
Response.Write oXML.XML
Set oXML = Nothing
%>
```

644 DOMDocument.CreateTextNode Method

Creates a new text node with the given string value.

644.1 Syntax

```
Set oTextNode = objXML.CreateTextNode(text)
```

644.2 Parameters

Parameter	Type	Required	Description
`text`	String	Yes	The text content of the new node.

644.3 Return Value

XMLElement. A new text node (NodeName = `#text`) that is not yet attached to any parent. Call `AppendChild` on the target element to attach it.

644.4 Remarks

- Method names are case-insensitive.

644.5 Code Example

```
<%
Dim oXML, oElem, oText
Set oXML = Server.CreateObject("MSXML2.DOMDocument")
Set oElem = oXML.CreateElement("title")
```

```
Set oText = oXML.CreateTextNode("My Title")
oElem.AppendChild oText
oXML.AppendChild oElem
Response.Write oXML.XML
Set oXML = Nothing
%>
```

645 DOMDocument.CreateAttribute Method

Creates a new, unattached attribute node with the given name.

645.1 Syntax

```
Set oAttr = objXML.CreateAttribute(name)
```

645.2 Parameters

Parameter	Type	Required	Description
name	String	Yes	The name of the attribute.

645.3 Return Value

XMLElement. A new attribute node not yet attached to any element. Set its `NodeValue` property to assign the attribute value, then use `SetAttribute` on the target element.

645.4 Remarks

- Method names are case-insensitive.

645.5 Code Example

```
<%
Dim oXML, oElem
Set oXML = Server.CreateObject("MSXML2.DOMDocument")
Set oElem = oXML.CreateElement("item")
oElem.SetAttribute "id", "42"
oXML.AppendChild oElem
Response.Write oXML.XML
Set oXML = Nothing
%>
```

646 DOMDocument.AppendChild Method

Attaches a child element to the document root. If the document has no root yet, the element becomes the root; otherwise it is appended as a child of the existing root.

646.1 Syntax

```
Set oResult = objXML.AppendChild(child)
```

646.2 Parameters

Parameter	Type	Required	Description
`child`	XMLElement	Yes	The element node to attach to the document.

646.3 Return Value

XMLElement. Returns the appended `child` node.

646.4 Remarks

- To build a complete document tree, call `CreateElement` on each node, build the sub-tree with `XMLElement.AppendChild`, then call `DOMDocument.AppendChild` with the root element.
- Method names are case-insensitive.

646.5 Code Example

```
<%
Dim oXML, oRoot, oItem
Set oXML = Server.CreateObject("MSXML2.DOMDocument")
Set oRoot = oXML.CreateElement("products")
Set oItem = oXML.CreateElement("product")
oItem.Text = "Widget"
oRoot.AppendChild oItem
oXML.AppendChild oRoot
Response.Write oXML.XML
Set oXML = Nothing
%>
```

647 DOMDocument.SelectSingleNode Method

Evaluates an XPath expression against the document and returns the first matching node.

647.1 Syntax

```
Set oNode = objXML.SelectSingleNode(xpath)
```

647.2 Parameters

Parameter	Type	Required	Description
`xpath`	String	Yes	An XPath 1.0 expression to evaluate against the document.

647.3 Return Value

XMLElement. The first node that satisfies the XPath expression. Returns Null if no node matches or the expression is empty.

647.4 Remarks

- XPath is evaluated from the document root.
- The `//` axis, attribute predicates (`@attr`), positional predicates, `contains()`, `starts-with()`, `not()`, and namespace-qualified expressions (when `SelectionNamespaces` is set) are all supported.
- Method names are case-insensitive.

647.5 Code Example

```
<%
Dim oXML, oNode
Set oXML = Server.CreateObject("MSXML2.DOMDocument")
oXML.LoadXML "<users><user id='1'>Alice</user><user id='2'>Bob</user></us
ers>"
Set oNode = oXML.SelectSingleNode("//user[@id='2']")
If Not IsNull(oNode) Then
    Response.Write oNode.Text
End If
Set oXML = Nothing
%>
```

648 DOMDocument.SelectNodes Method

Evaluates an XPath expression against the document and returns all matching nodes.

648.1 Syntax

```
Set oList = objXML.SelectNodes(xpath)
```

648.2 Parameters

Parameter	Type	Required	Description
`xpath`	String	Yes	An XPath 1.0 expression to evaluate against the document.

648.3 Return Value

XMLNodeList. All nodes that satisfy the XPath expression in document order. Returns an empty XMLNodeList if no nodes match or the expression is empty.

648.4 Remarks

- XPath is evaluated from the document root.
- The `//` axis, attribute predicates, positional predicates, `contains()`, `starts-with()`, `not()`, `and`, `or`, and namespace-qualified expressions are all supported.
- Use `SelectionNamespaces` on the document to bind namespace prefixes before querying.
- Method names are case-insensitive.

648.5 Code Example

```
<%
Dim oXML, oList, i
Set oXML = Server.CreateObject("MSXML2.DOMDocument")
oXML.LoadXML "<items><item cat='A'>One</item><item cat='B'>Two</item><ite
m cat='A'>Three</item></items>"
Set oList = oXML.SelectNodes("//item[@cat='A']")
For i = 0 To oList.Length - 1
    Response.Write oList.Item(i).Text & "<br>"
Next
Set oXML = Nothing
%>
```

649 XMLNodeList.Item Method

Returns the node at a given zero-based index from the list.

649.1 Syntax

```
Set oNode = oList.Item(index)
```

649.2 Parameters

Parameter	Type	Required	Description
`index`	Integer	Yes	Zero-based position of the node to retrieve.

649.3 Return Value

XMLElement. The node at the specified index. Returns Null if the index is negative or greater than or equal to `Length`.

649.4 Remarks

- Method names are case-insensitive.

649.5 Code Example

```
<%
Dim oXML, oList, oNode
Set oXML = Server.CreateObject("MSXML2.DOMDocument")
oXML.LoadXML "<list><a>1</a><b>2</b><c>3</c></list>"
Set oList = oXML.DocumentElement.ChildNodes
Set oNode = oList.Item(1)
Response.Write oNode.NodeName & ": " & oNode.Text
Set oXML = Nothing
%>
```

650 XMLNodeList.NextNode Method

Returns the next node in a forward-only iteration over the list.

650.1 Syntax

```
Set oNode = oList.NextNode()
```

650.2 Parameters

None.

650.3 Return Value

XMLElement. The next node in iteration order. Returns Null when all nodes have been visited.

650.4 Remarks

- The iterator position is maintained internally within the XMLNodeList object.
- Method names are case-insensitive.

650.5 Code Example

```
<%
Dim oXML, oList, oNode
Set oXML = Server.CreateObject("MSXML2.DOMDocument")
oXML.LoadXML "<items><item>A</item><item>B</item><item>C</item></items>"
Set oList = oXML.GetElementsByTagName("item")
Set oNode = oList.NextNode()
Do While Not IsNull(oNode)
    Response.Write oNode.Text & "<br>"
    Set oNode = oList.NextNode()
Loop
Set oXML = Nothing
%>
```

651 XMLElement.AppendChild Method

Appends a child node to this element and returns the appended node.

651.1 Syntax

```
Set oResult = oElement.AppendChild(child)
```

651.2 Parameters

Parameter	Type	Required	Description
`child`	XMLElement	Yes	The node to append as a direct child of this element.

651.3 Return Value

XMLElement. Returns the appended `child` node.

651.4 Remarks

- Method names are case-insensitive.

651.5 Code Example

```
<%
Dim oXML, oParent, oChild
Set oXML = Server.CreateObject("MSXML2.DOMDocument")
oXML.LoadXML "<root></root>"
Set oParent = oXML.DocumentElement
Set oChild = oXML.CreateElement("entry")
oChild.Text = "New entry"
oParent.AppendChild oChild
Response.Write oXML.XML
Set oXML = Nothing
%>
```

652 XMLElement.GetElementsByTagName Method

Returns all descendant elements of this element that match the given tag name.

652.1 Syntax

```
Set oList = oElement.GetElementsByTagName(tagName)
```

652.2 Parameters

Parameter	Type	Required	Description
`tagName`	String	Yes	The tag name to

Parameter	Type	Required	Description
			search for. The match is case-insensitive.

652.3 Return Value

XMLNodeList. All matching descendant elements in document order. Returns an empty XMLNodeList if no elements match.

652.4 Remarks

- The search includes all levels of descendants, not just direct children.
- Method names are case-insensitive.

652.5 Code Example

```
<%
Dim oXML, oRoot, oList, i
Set oXML = Server.CreateObject("MSXML2.DOMDocument")
oXML.LoadXML "<root><section><item>A</item></section><item>B</item></root>
"
Set oRoot = oXML.DocumentElement
Set oList = oRoot.GetElementsByTagName("item")
For i = 0 To oList.Length - 1
    Response.Write oList.Item(i).Text & "<br>"
Next
Set oXML = Nothing
%>
```

653 XMLElement.Item Method

Returns the direct child node at a given zero-based index.

653.1 Syntax

```
Set oChild = oElement.Item(index)
```

653.2 Parameters

Parameter	Type	Required	Description
index	Integer	Yes	Zero-based position among the direct children of this element.

653.3 Return Value

XMLElement. The child node at the specified index. Returns Null if the index is out of range.

653.4 Remarks

- Method names are case-insensitive.

653.5 Code Example

```
<%
Dim oXML, oRoot, oChild
Set oXML = Server.CreateObject("MSXML2.DOMDocument")
oXML.LoadXML "<root><a>1</a><b>2</b></root>"
Set oRoot = oXML.DocumentElement
Set oChild = oRoot.Item(0)
Response.Write oChild.NodeName & ": " & oChild.Text
Set oXML = Nothing
%>
```

654 XMLElement.SetAttribute Method

Creates or replaces an attribute on this element.

654.1 Syntax

```
oElement.SetAttribute name, value
```

654.2 Parameters

Parameter	Type	Required	Description
`name`	String	Yes	The attribute name.
`value`	String	Yes	The value to assign to the attribute.

654.3 Return Value

Empty. This method does not return a value.

654.4 Remarks

- If an attribute with the same name already exists, its value is replaced.
- Method names are case-insensitive.

654.5 Code Example

```
<%
Dim oXML, oElem
Set oXML = Server.CreateObject("MSXML2.DOMDocument")
Set oElem = oXML.CreateElement("product")
oElem.SetAttribute "id", "101"
```

```
oElem.SetAttribute "category", "electronics"
oElem.Text = "Gadget"
oXML.AppendChild oElem
Response.Write oXML.XML
Set oXML = Nothing
%>
```

655 XMLElement.GetAttribute Method

Returns the value of the named attribute on this element.

655.1 Syntax

```
value = oElement.GetAttribute(name)
```

655.2 Parameters

Parameter	Type	Required	Description
name	String	Yes	The attribute name to look up.

655.3 Return Value

String. The value of the named attribute. Returns an empty String if the attribute does not exist.

655.4 Remarks

- Method names are case-insensitive.

655.5 Code Example

```
<%
Dim oXML, oNode
Set oXML = Server.CreateObject("MSXML2.DOMDocument")
oXML.LoadXML "<item id='99' name='Widget'/>"
Set oNode = oXML.DocumentElement
Response.Write "id=" & oNode.GetAttribute("id") & ", name=" & oNode.GetAt
tribute("name")
Set oXML = Nothing
%>
```

656 XMLElement.RemoveAttribute Method

Removes the named attribute from this element.

656.1 Syntax

```
oElement.RemoveAttribute name
```

656.2 Parameters

Parameter	Type	Required	Description
name	String	Yes	The name of the attribute to remove.

656.3 Return Value

Empty. This method does not return a value.

656.4 Remarks

- If the attribute does not exist, the call is silently ignored.
- Method names are case-insensitive.

656.5 Code Example

```
<%
Dim oXML, oNode
Set oXML = Server.CreateObject("MSXML2.DOMDocument")
oXML.LoadXML "<item id='1' temp='yes'/>"
Set oNode = oXML.DocumentElement
oNode.RemoveAttribute "temp"
Response.Write oXML.XML
Set oXML = Nothing
%>
```

657 XMLElement.SelectSingleNode Method

Evaluates an XPath expression relative to this element and returns the first matching node.

657.1 Syntax

```
Set oNode = oElement.SelectSingleNode(xpath)
```

657.2 Parameters

Parameter	Type	Required	Description
xpath	String	Yes	An XPath 1.0 expression evaluated relative to this element.

657.3 Return Value

XMLElement. The first matching node. Returns Null if no node matches or the expression is empty.

657.4 Remarks

- When the XPath expression starts with `/`, it is evaluated from the document root rather than the current element.
- Method names are case-insensitive.

657.5 Code Example

```
<%
Dim oXML, oSection, oNode
Set oXML = Server.CreateObject("MSXML2.DOMDocument")
oXML.LoadXML "<root><section><item id='3'>Target</item></section></root>"
Set oSection = oXML.SelectSingleNode("//section")
Set oNode = oSection.SelectSingleNode("item[@id='3']")
If Not IsNull(oNode) Then
    Response.Write oNode.Text
End If
Set oXML = Nothing
%>
```

658 XMLElement.SelectNodes Method

Evaluates an XPath expression relative to this element and returns all matching nodes.

658.1 Syntax

```
Set oList = oElement.SelectNodes(xpath)
```

658.2 Parameters

Parameter	Type	Required	Description
`xpath`	String	Yes	An XPath 1.0 expression evaluated relative to this element.

658.3 Return Value

XMLNodeList. All matching nodes in document order. Returns an empty XMLNodeList if no nodes match or the expression is empty.

658.4 Remarks

- When the XPath expression starts with `/`, it is evaluated from the document root.
- Method names are case-insensitive.

658.5 Code Example

```
<%
Dim oXML, oSection, oList, i
Set oXML = Server.CreateObject("MSXML2.DOMDocument")
oXML.LoadXML "<root><group><item>A</item><item>B</item></group></root>"
Set oSection = oXML.SelectSingleNode("//group")
Set oList = oSection.SelectNodes("item")
For i = 0 To oList.Length - 1
    Response.Write oList.Item(i).Text & "<br>"
Next
Set oXML = Nothing
%>
```

659 ServerXMLHTTP.ResponseText Property

Returns the response body decoded as a text string.

659.1 Access

Read-only.

659.2 Type

String.

659.3 Remarks

- Available after `Send` completes (`ReadyState = 4`).
- The response body is decoded using the charset specified by the server's `Content-Type` response header. Falls back to UTF-8 if no charset is present.
- Returns an empty String if no request has been sent or the body is empty.

659.4 Code Example

```
<%
Dim oHTTP
Set oHTTP = Server.CreateObject("MSXML2.ServerXMLHTTP")
oHTTP.Open "GET", "https://example.com/data.txt", False
oHTTP.Send
If oHTTP.Status = 200 Then
    Response.Write oHTTP.ResponseText
End If
Set oHTTP = Nothing
%>
```

660 ServerXMLHTTP.ResponseXML Property

Returns the response body parsed as an XML document.

660.1 Access

Read-only.

660.2 Type

DOMDocument (if the response is valid XML) or String (if parsing fails).

660.3 Remarks

- Available after `Send` completes (`ReadyState = 4`).
- If the response body is valid XML, a fully populated DOMDocument object is returned. You can call `SelectSingleNode`, `GetElementsByTagName`, and access `DocumentElement` on it.
- If the response body cannot be parsed as XML, the raw response text is returned as a String.
- Returns an empty String if no response has been received.

660.4 Code Example

```
<%
Dim oHTTP, oXML, oNode
Set oHTTP = Server.CreateObject("MSXML2.ServerXMLHTTP")
oHTTP.Open "GET", "https://example.com/feed.xml", False
oHTTP.Send
If oHTTP.Status = 200 Then
    Set oXML = oHTTP.ResponseXML
    Set oNode = oXML.SelectSingleNode("//title")
    If Not IsNull(oNode) Then
        Response.Write oNode.Text
    End If
End If
Set oHTTP = Nothing
%>
```

661 ServerXMLHTTP.ResponseBody Property

Returns the raw response body as a byte array.

661.1 Access

Read-only.

661.2 Type

Byte Array.

661.3 Remarks

- Available after `Send` completes (`ReadyState = 4`).
- The returned array contains the unprocessed response bytes with no character decoding applied.
- Useful for downloading binary content such as images, PDFs, or compressed files.
- Returns an empty array if no response has been received.

661.4 Code Example

```
<%
Dim oHTTP
Set oHTTP = Server.CreateObject("MSXML2.ServerXMLHTTP")
oHTTP.Open "GET", "https://example.com/file.zip", False
oHTTP.Send
If oHTTP.Status = 200 Then
    Response.BinaryWrite oHTTP.ResponseBody
End If
Set oHTTP = Nothing
%>
```

662 ServerXMLHTTP.Status Property

Returns the HTTP status code from the response.

662.1 Access

Read-only.

662.2 Type

Integer.

662.3 Remarks

- Available after `Send` completes (`ReadyState = 4`).
- Common values: 200 (OK), 404 (Not Found), 500 (Internal Server Error).
- Returns 0 if no request has been sent or if the connection failed before any response was received.

662.4 Code Example

```
<%
Dim oHTTP
Set oHTTP = Server.CreateObject("MSXML2.ServerXMLHTTP")
oHTTP.Open "GET", "https://example.com/page", False
oHTTP.Send
Select Case oHTTP.Status
```

```
    Case 200
        Response.Write "OK: " & oHTTP.ResponseText
    Case 404
        Response.Write "Not found."
    Case Else
        Response.Write "HTTP " & oHTTP.Status & ": " & oHTTP.StatusText
End Select
Set oHTTP = Nothing
%>
```

663 ServerXMLHTTP.StatusText Property

Returns the full HTTP status line from the response.

663.1 Access

Read-only.

663.2 Type

String.

663.3 Remarks

- Available after `Send` completes (`ReadyState = 4`).
- The value includes the status code and reason phrase, for example `200 OK` or `404 Not Found`.
- Returns an empty String if no response has been received.

663.4 Code Example

```
<%
Dim oHTTP
Set oHTTP = Server.CreateObject("MSXML2.ServerXMLHTTP")
oHTTP.Open "GET", "https://example.com/", False
oHTTP.Send
Response.Write "Status: " & oHTTP.StatusText
Set oHTTP = Nothing
%>
```

664 ServerXMLHTTP.ReadyState Property

Returns the current lifecycle state of the HTTP request.

664.1 Access

Read-only.

664.2 Type

Integer.

664.3 Values

Value	Name	Description
0	Uninitialized	The object has been created but Open has not been called.
1	Open	Open has been called but Send has not been called.
2	Sent	Send has been called but no response has been received yet.
3	Receiving	Response headers have been received; body is being received.
4	Complete	The response is fully received and available.

664.4 Remarks

- Because the implementation is synchronous, ReadyState transitions directly from 1 to 4 when Send returns.

664.5 Code Example

```
<%
Dim oHTTP
Set oHTTP = Server.CreateObject("MSXML2.ServerXMLHTTP")
Response.Write "Before Open: " & oHTTP.ReadyState & "<br>"
oHTTP.Open "GET", "https://example.com/", False
Response.Write "After Open: " & oHTTP.ReadyState & "<br>"
oHTTP.Send
Response.Write "After Send: " & oHTTP.ReadyState & "<br>"
Set oHTTP = Nothing
%>
```

665 ServerXMLHTTP.Timeout Property

Gets or sets the maximum number of seconds to wait for a response before the request fails.

665.1 Access

Read/Write.

665.2 Type

Integer.

665.3 Default

30 seconds.

665.4 Remarks

- Set this property before calling `Send`.
- If the server does not respond within the configured time, `Send` returns with an error condition and `Status` is set to 0.
- Setting `Timeout` to 0 is not recommended as it may cause indefinite blocking.

665.5 Code Example

```
<%
Dim oHTTP
Set oHTTP = Server.CreateObject("MSXML2.ServerXMLHTTP")
oHTTP.Timeout = 10
oHTTP.Open "GET", "https://example.com/slow-endpoint", False
On Error Resume Next
oHTTP.Send
If Err.Number <> 0 Then
    Response.Write "Request timed out."
    Err.Clear
Else
    Response.Write oHTTP.ResponseText
End If
Set oHTTP = Nothing
%>
```

666 DOMDocument.DocumentElement Property

Returns the root element of the parsed document.

666.1 Access

Read-only.

666.2 Type

XMLElement or Null.

666.3 Remarks

- Returns Null if the document has not been loaded, or if `LoadXML` or `Load` failed.
- The returned XMLElement is the top-level element of the document tree.

666.4 Code Example

```
<%
Dim oXML, oRoot
Set oXML = Server.CreateObject("MSXML2.DOMDocument")
oXML.LoadXML "<catalog><item>First</item></catalog>"
Set oRoot = oXML.DocumentElement
If Not IsNull(oRoot) Then
    Response.Write "Root element: " & oRoot.NodeName
End If
Set oXML = Nothing
%>
```

667 DOMDocument.XML Property

Returns the XML source string or the serialized representation of the current document tree.

667.1 Access

Read-only.

667.2 Type

String.

667.3 Remarks

- If the document was loaded with `LoadXML`, the original XML string is returned.
- If the document was built programmatically using `CreateElement` and `AppendChild`, the tree is serialized to an XML string.
- Returns an empty String if no document has been loaded.

667.4 Code Example

```
<%
Dim oXML, oRoot, oItem
Set oXML = Server.CreateObject("MSXML2.DOMDocument")
Set oRoot = oXML.CreateElement("list")
Set oItem = oXML.CreateElement("entry")
oItem.Text = "Hello"
oRoot.AppendChild oItem
oXML.AppendChild oRoot
Response.Write oXML.XML
```

```
Set oXML = Nothing
%>
```

668 DOMDocument.ParseError Property

Returns the ParseError object that describes the outcome of the last `LoadXML` or `Load` call.

668.1 Access

Read-only.

668.2 Type

ParseError object.

668.3 Remarks

- This property always returns a ParseError object, even when parsing succeeded.
- After a successful parse, `ParseError.ErrorCode` is 0.
- After a failed parse, `ParseError` is populated with the error details, including `Reason`, `Line`, `LinePos`, and `SrcText`.

668.4 Code Example

```
<%
Dim oXML
Set oXML = Server.CreateObject("MSXML2.DOMDocument")
If Not oXML.LoadXML("<root><unclosed>") Then
    Response.Write "Error " & oXML.ParseError.ErrorCode & ": " & oXML.Par
seError.Reason
    Response.Write " (Line " & oXML.ParseError.Line & ", Col " & oXML.Par
seError.LinePos & ")"
End If
Set oXML = Nothing
%>
```

669 DOMDocument.Async Property

Gets or sets whether the document loads asynchronously.

669.1 Access

Read/Write.

669.2 Type

Boolean.

669.3 Default

False.

669.4 Remarks

- This property is accepted for compatibility with existing code. The current AxonASP implementation always loads documents synchronously regardless of this setting.
- Set to False before calling `Load` or `LoadXML` to follow the Classic ASP convention.

669.5 Code Example

```
<%
Dim oXML
Set oXML = Server.CreateObject("MSXML2.DOMDocument")
oXML.Async = False
oXML.LoadXML "<data/>"
Response.Write oXML.DocumentElement.NodeName
Set oXML = Nothing
%>
```

670 DOMDocument.ServerHTTPRequest Property

Gets or sets whether HTTP requests issued by the document use the server HTTP stack.

670.1 Access

Read/Write.

670.2 Type

Boolean.

670.3 Default

False.

670.4 Remarks

- This property controls the HTTP client stack used when `Load` fetches content from an HTTP or HTTPS URL.
- Accepted for compatibility; the current implementation uses the same HTTP client regardless of this setting.

670.5 Code Example

```
<%
Dim oXML
Set oXML = Server.CreateObject("MSXML2.DOMDocument")
oXML.ServerHTTPRequest = True
oXML.Load "https://example.com/data.xml"
Set oXML = Nothing
%>
```

671 DOMDocument.ResolveExternals Property

Gets or sets whether external entities and document type definitions are resolved during parsing.

671.1 Access

Read/Write.

671.2 Type

Boolean.

671.3 Default

False.

671.4 Remarks

- Accepted for compatibility with existing code. Set this property before calling `Load` or `LoadXML`.

671.5 Code Example

```
<%
Dim oXML
Set oXML = Server.CreateObject("MSXML2.DOMDocument")
oXML.ResolveExternals = False
oXML.LoadXML "<root/>"
Set oXML = Nothing
%>
```

672 DOMDocument.ValidateOnParse Property

Gets or sets whether the document is validated against its DTD or schema during parsing.

672.1 Access

Read/Write.

672.2 Type

Boolean.

672.3 Default

False.

672.4 Remarks

- Accepted for compatibility with existing code. Set this property before calling `Load` or `LoadXML`.

672.5 Code Example

```
<%
Dim oXML
Set oXML = Server.CreateObject("MSXML2.DOMDocument")
oXML.ValidateOnParse = False
oXML.LoadXML "<root/>"
Set oXML = Nothing
%>
```

673 DOMDocument.PreserveWhiteSpace Property

Gets or sets whether insignificant whitespace text nodes are preserved in the document tree.

673.1 Access

Read/Write.

673.2 Type

Boolean.

673.3 Default

False.

673.4 Remarks

- When False, whitespace-only text nodes between elements are discarded during parsing.
- Set this property before calling `Load` or `LoadXML`.

673.5 Code Example

```
<%
Dim oXML
Set oXML = Server.CreateObject("MSXML2.DOMDocument")
```

```
oXML.PreserveWhiteSpace = False
oXML.LoadXML "<root>  <item>A</item>  </root>"
Response.Write oXML.DocumentElement.ChildNodes.Length
Set oXML = Nothing
%>
```

674 DOMDocument.SelectionLanguage Property

Gets or sets the query language used by `SelectSingleNode` and `SelectNodes`.

674.1 Access

Read/Write.

674.2 Type

String.

674.3 Default

`"XPath"`

674.4 Remarks

- The only currently supported value is `XPath`.
- Setting this property is equivalent to calling `SetProperty "SelectionLanguage", value`.

674.5 Code Example

```
<%
Dim oXML
Set oXML = Server.CreateObject("MSXML2.DOMDocument")
oXML.SelectionLanguage = "XPath"
oXML.LoadXML "<root><item>A</item></root>"
Response.Write oXML.SelectSingleNode("//item").Text
Set oXML = Nothing
%>
```

675 DOMDocument.SelectionNamespaces Property

Gets or sets the namespace prefix bindings used during XPath queries.

675.1 Access

Read/Write.

675.2 Type

String.

675.3 Remarks

- The value is a space-separated list of namespace declarations in the format `xmlns:prefix='uri'`.
- Setting this property is equivalent to calling `SetProperty "SelectionNamespaces", value`.
- Must be set before calling `SelectSingleNode` or `SelectNodes` on namespace-qualified documents.

675.4 Code Example

```
<%
Dim oXML, oNode
Set oXML = Server.CreateObject("MSXML2.DOMDocument")
oXML.LoadXML "<ns:catalog xmlns:ns='urn:example'><ns:item>Widget</ns:item>
</ns:catalog>"
oXML.SelectionNamespaces = "xmlns:ns='urn:example'"
Set oNode = oXML.SelectSingleNode("//ns:item")
If Not IsNull(oNode) Then
    Response.Write oNode.Text
End If
Set oXML = Nothing
%>
```

676 XMLNodeList.Length Property

Returns the number of nodes in the list.

676.1 Access

Read-only.

676.2 Type

Integer.

676.3 Remarks

- Use this property to iterate the list with `Item(0)` through `Item(Length - 1)`.
- Returns o for an empty list.

676.4 Code Example

```
<%
Dim oXML, oList, i
Set oXML = Server.CreateObject("MSXML2.DOMDocument")
oXML.LoadXML "<r><a/><b/><c/></r>"
Set oList = oXML.DocumentElement.ChildNodes
For i = 0 To oList.Length - 1
    Response.Write oList.Item(i).NodeName & "<br>"
```

```
Next
Set oXML = Nothing
%>
```

677 XMLNodeList.Count Property

Returns the number of nodes in the list. This is an alias for `Length`.

677.1 Access

Read-only.

677.2 Type

Integer.

677.3 Remarks

- Identical in behaviour to `Length`. Either property can be used interchangeably.
- Returns 0 for an empty list.

677.4 Code Example

```
<%
Dim oXML, oList
Set oXML = Server.CreateObject("MSXML2.DOMDocument")
oXML.LoadXML "<r><a/><b/></r>"
Set oList = oXML.DocumentElement.ChildNodes
Response.Write "Count: " & oList.Count
Set oXML = Nothing
%>
```

678 ParseError.ErrorCode Property

Returns the numeric error code from the last parse operation.

678.1 Access

Read-only.

678.2 Type

Integer.

678.3 Remarks

- A value of 0 indicates the document was parsed successfully.
- A negative value (typically -1) indicates a general parse or file-load failure.
- When `DOMDocument.Load` fails due to an HTTP error, this property is set to the HTTP status code (e.g., 404, 500).

- Always check this property after `LoadXML` or `Load` returns False before accessing document content.

678.4 Code Example

```
<%
Dim oXML
Set oXML = Server.CreateObject("MSXML2.DOMDocument")
If Not oXML.LoadXML("<bad<xml") Then
    Response.Write "Parse failed. ErrorCode: " & oXML.ParseError.ErrorCod
e
End If
Set oXML = Nothing
%>
```

679 ParseError.Reason Property

Returns a human-readable description of the parse error.

679.1 Access

Read-only.

679.2 Type

String.

679.3 Remarks

- Returns an empty String when `ErrorCode` is 0 (no error).
- For HTTP load failures, the reason string describes the network or server error.

679.4 Code Example

```
<%
Dim oXML
Set oXML = Server.CreateObject("MSXML2.DOMDocument")
If Not oXML.LoadXML("<open") Then
    Response.Write "Reason: " & oXML.ParseError.Reason
End If
Set oXML = Nothing
%>
```

680 ParseError.FilePos Property

Returns the byte offset in the source document where the parse error was detected.

680.1 Access

Read-only.

680.2 Type

Integer.

680.3 Remarks

- Returns 0 when there is no error or when the position cannot be determined.
- Use `Line` and `LinePos` for line-and-column positioning.

680.4 Code Example

```
<%
Dim oXML
Set oXML = Server.CreateObject("MSXML2.DOMDocument")
If Not oXML.LoadXML("<a><b></a>") Then
    Response.Write "Error at byte offset: " & oXML.ParseError.FilePos
End If
Set oXML = Nothing
%>
```

681 ParseError.Line Property

Returns the 1-based line number in the source document where the parse error was detected.

681.1 Access

Read-only.

681.2 Type

Integer.

681.3 Remarks

- Returns 0 when there is no error.
- Pair with `LinePos` to identify the exact error location.

681.4 Code Example

```
<%
Dim oXML
Set oXML = Server.CreateObject("MSXML2.DOMDocument")
If Not oXML.LoadXML("<root>" & Chr(10) & "<bad attr") Then
    Response.Write "Error on line: " & oXML.ParseError.Line
End If
Set oXML = Nothing
%>
```

682 ParseError.LinePos Property

Returns the 1-based column position within the error line where the parse error was detected.

682.1 Access

Read-only.

682.2 Type

Integer.

682.3 Remarks

- Returns 0 when there is no error.
- Pair with `Line` for a precise error location.

682.4 Code Example

```
<%
Dim oXML
Set oXML = Server.CreateObject("MSXML2.DOMDocument")
If Not oXML.LoadXML("<root><bad") Then
    Response.Write "Line " & oXML.ParseError.Line & ", Col " & oXML.Parse
Error.LinePos
End If
Set oXML = Nothing
%>
```

683 ParseError.SrcText Property

Returns the source text that was being parsed when the error occurred.

683.1 Access

Read-only.

683.2 Type

String.

683.3 Remarks

- Contains the text fragment around the error position.
- Returns an empty String when there is no error.

683.4 Code Example

```
<%
Dim oXML
```

```
Set oXML = Server.CreateObject("MSXML2.DOMDocument")
If Not oXML.LoadXML("<root><item broken") Then
    Response.Write "Near: " & oXML.ParseError.SrcText
End If
Set oXML = Nothing
%>
```

684 ParseError.URL Property

Returns the URL or file path that was being loaded when the error occurred.

684.1 Access

Read-only.

684.2 Type

String.

684.3 Remarks

- Populated when `Load` fails, and contains the URL or resolved file path that was requested.
- Returns an empty String when the document was loaded via `LoadXML` or when there is no error.

684.4 Code Example

```
<%
Dim oXML
Set oXML = Server.CreateObject("MSXML2.DOMDocument")
If Not oXML.Load("https://example.com/missing.xml") Then
    Response.Write "Failed to load: " & oXML.ParseError.URL
    Response.Write " (" & oXML.ParseError.Reason & ")"
End If
Set oXML = Nothing
%>
```

685 XMLElement.NodeName Property

Returns the tag name of the element node.

685.1 Access

Read-only.

685.2 Type

String.

685.3 Remarks

- For element nodes, this is the XML tag name (e.g., `"product"`, `"item"`).
- For text nodes created with `CreateTextNode`, the value is `"#text"`.
- For attribute nodes created with `CreateAttribute`, the value is the attribute name.

685.4 Code Example

```
<%
Dim oXML, oNode
Set oXML = Server.CreateObject("MSXML2.DOMDocument")
oXML.LoadXML "<catalog><item>Widget</item></catalog>"
Set oNode = oXML.SelectSingleNode("//item")
Response.Write oNode.NodeName
Set oXML = Nothing
%>
```

686 XMLElement.NodeValue Property

Gets or sets the value of the node.

686.1 Access

Read/Write.

686.2 Type

String.

686.3 Remarks

- For text nodes (`NodeName = "#text"`), this property holds the text content.
- For attribute nodes, this holds the attribute value.
- For element nodes, this property is typically empty; use the `Text` property to get concatenated text content.
- Setting `NodeValue` replaces the node's current value.

686.4 Code Example

```
<%
Dim oXML, oNode
Set oXML = Server.CreateObject("MSXML2.DOMDocument")
oXML.LoadXML "<msg>Hello</msg>"
Set oNode = oXML.DocumentElement.FirstChild
Response.Write "NodeValue: " & oNode.NodeValue
Set oXML = Nothing
%>
```

687 XMLElement.Text Property

Gets or sets the concatenated text content of the element and all its descendants.

687.1 Access

Read/Write.

687.2 Type

String.

687.3 Remarks

- When read, the property traverses the full subtree and concatenates all text node values.
- When assigned, the element's child nodes are replaced with a single new text node containing the given string.
- For a node with no text content, returns an empty String.

687.4 Code Example

```
<%
Dim oXML, oNode
Set oXML = Server.CreateObject("MSXML2.DOMDocument")
oXML.LoadXML "<root><a>Hello </a><b>World</b></root>"
Set oNode = oXML.DocumentElement
Response.Write oNode.Text
Set oXML = Nothing
%>
```

688 XMLElement.XML Property

Returns the serialized XML markup of this node and its entire subtree.

688.1 Access

Read-only.

688.2 Type

String.

688.3 Remarks

- The returned string includes the element's opening tag, all child nodes serialized recursively, and the closing tag.
- Attributes on the element are included in the opening tag.
- Use `DOMDocument.XML` to serialize the complete document.

688.4Code Example

```
<%
Dim oXML, oNode
Set oXML = Server.CreateObject("MSXML2.DOMDocument")
oXML.LoadXML "<root><child id='1'>Content</child></root>"
Set oNode = oXML.SelectSingleNode("//child")
Response.Write oNode.XML
Set oXML = Nothing
%>
```

689 XMLElement.Attributes Property

Returns a collection of the element's attribute name/value pairs.

689.1Access

Read-only.

689.2Type

Collection (array of name/value maps).

689.3Remarks

- Each entry in the collection represents one attribute.
- Use `GetAttribute` and `SetAttribute` for direct attribute read/write by name.
- The order of attributes in the collection is not guaranteed.

689.4Code Example

```
<%
Dim oXML, oNode
Set oXML = Server.CreateObject("MSXML2.DOMDocument")
oXML.LoadXML "<item id='5' name='Widget'/>"
Set oNode = oXML.DocumentElement
Response.Write "id=" & oNode.GetAttribute("id") & ", name=" & oNode.GetAt
tribute("name")
Set oXML = Nothing
%>
```

690 XMLElement.ChildNodes Property

Returns a list of all direct child nodes of this element.

690.1Access

Read-only.

690.2 Type

XMLNodeList.

690.3 Remarks

- The list contains only direct children, not descendants at deeper levels.
- Returns an empty XMLNodeList if the element has no children.
- Use `Children` for an identical result.

690.4 Code Example

```
<%
Dim oXML, oRoot, oList, i
Set oXML = Server.CreateObject("MSXML2.DOMDocument")
oXML.LoadXML "<root><a/><b/><c/></root>"
Set oRoot = oXML.DocumentElement
Set oList = oRoot.ChildNodes
For i = 0 To oList.Length - 1
    Response.Write oList.Item(i).NodeName & "<br>"
Next
Set oXML = Nothing
%>
```

691 XMLElement.FirstChild Property

Returns the first direct child node of this element.

691.1 Access

Read-only.

691.2 Type

XMLElement or Null.

691.3 Remarks

- Returns Null if the element has no children.

691.4 Code Example

```
<%
Dim oXML, oRoot, oFirst
Set oXML = Server.CreateObject("MSXML2.DOMDocument")
oXML.LoadXML "<root><a>First</a><b>Second</b></root>"
Set oRoot = oXML.DocumentElement
Set oFirst = oRoot.FirstChild
If Not IsNull(oFirst) Then
    Response.Write oFirst.Text
End If
```

```
Set oXML = Nothing
%>
```

692 XMLElement.LastChild Property

Returns the last direct child node of this element.

692.1 Access

Read-only.

692.2 Type

XMLElement or Null.

692.3 Remarks

- Returns Null if the element has no children.

692.4 Code Example

```
<%
Dim oXML, oRoot, oLast
Set oXML = Server.CreateObject("MSXML2.DOMDocument")
oXML.LoadXML "<root><a>First</a><b>Second</b></root>"
Set oRoot = oXML.DocumentElement
Set oLast = oRoot.LastChild
If Not IsNull(oLast) Then
    Response.Write oLast.Text
End If
Set oXML = Nothing
%>
```

693 XMLElement.ParentNode Property

Returns the parent element of this node.

693.1 Access

Read-only.

693.2 Type

XMLElement or Null.

693.3 Remarks

- Returns Null if this node is the document root or has not been attached to a tree.

693.4 Code Example

```
<%
Dim oXML, oChild
Set oXML = Server.CreateObject("MSXML2.DOMDocument")
oXML.LoadXML "<root><section><item>Test</item></section></root>"
Set oChild = oXML.SelectSingleNode("//item")
If Not IsNull(oChild.ParentNode) Then
    Response.Write "Parent: " & oChild.ParentNode.NodeName
End If
Set oXML = Nothing
%>
```

694 XMLElement.Length Property

Returns the number of direct child nodes of this element.

694.1 Access

Read-only.

694.2 Type

Integer.

694.3 Remarks

- Equivalent to `ChildNodes.Length`.
- Returns 0 if the element has no children.

694.4 Code Example

```
<%
Dim oXML, oRoot
Set oXML = Server.CreateObject("MSXML2.DOMDocument")
oXML.LoadXML "<root><a/><b/><c/></root>"
Set oRoot = oXML.DocumentElement
Response.Write "Children: " & oRoot.Length
Set oXML = Nothing
%>
```

695 XMLElement.Children Property

Returns a list of all direct child nodes of this element. This is an alias for `ChildNodes`.

695.1 Access

Read-only.

695.2 Type

XMLNodeList.

695.3 Remarks

- Identical in behaviour to `ChildNodes`. Either property can be used interchangeably.
- Returns an empty XMLNodeList if the element has no children.

695.4 Code Example

```
<%
Dim oXML, oRoot, oList
Set oXML = Server.CreateObject("MSXML2.DOMDocument")
oXML.LoadXML "<root><x/><y/></root>"
Set oRoot = oXML.DocumentElement
Set oList = oRoot.Children
Response.Write "Child count: " & oList.Length
Set oXML = Nothing
%>
```

696 ASP Example Applications

696.1 Overview

AxonASP ships with four example web applications located under the `www/` directory. Each example demonstrates a common web application pattern implemented in Classic ASP using AxonASP features. The examples are intended as learning references and starting points for new projects.

Access the examples from the web root while the HTTP server is running at `http://localhost:8801/`.

696.2 MVC — Model-View-Controller

Path: `./www/mvc/` **URL:** `http://localhost:8801/mvc/`

This example implements the Model-View-Controller pattern in Classic ASP. Responsibilities are separated into three layers:

Directory / File	Role
`mvc/model/site_model.asp`	Data and business logic functions
`mvc/controller/site_controller.asp`	Request handling, page routing,

Directory / File	Role
	assembling data for views
`mvc/view/`	HTML generation functions, one per template
`mvc/default.asp`	Front controller entry point

How it works: The front controller (`default.asp`) reads a page query string parameter, delegates to the appropriate controller function, and the controller calls the model and a view function to produce HTML output.

Example request:

```
http://localhost:8801/mvc/?page=home
http://localhost:8801/mvc/?page=example
```

696.3MVVM — Model-View-ViewModel

Path: `./www/mvvm/` **URL:** `http://localhost:8801/mvvm/`

This example implements the Model-View-ViewModel pattern in Classic ASP. The ViewModel layer prepares display-ready data and exposes it to the View, decoupling rendering from business logic.

Directory / File	Role
`mvvm/model/site_model.asp`	Data access and domain logic
`mvvm/viewmodel/site_viewmodel.asp`	Data transformation and preparation for display
`mvvm/view/`	HTML templates consuming ViewModel output
`mvvm/default.asp`	Entry point and routing

How it works: The entry point includes the model and viewmodel layers, resolves the requested route, calls the appropriate ViewModel function, and outputs the rendered HTML.

Example request:

```
http://localhost:8801/mvvm/?page=home
http://localhost:8801/mvvm/?page=example
```

696.4REST — Basic REST-Style Endpoints

Path: `./www/rest/` **URL:** `http://localhost:8801/rest/`

This example demonstrates basic REST-style HTTP endpoints using separate `.asp` files under an `api/` subdirectory. A browser front-end page makes JavaScript `fetch` requests to the endpoints and displays the JSON responses.

Path	Description
`rest/default.asp`	Demo front-end page with buttons to call the API endpoints
`rest/api/status.asp`	Returns a JSON status response
`rest/api/echo.asp`	Returns the `name` query string parameter as a JSON echo response

How it works: Each endpoint file in `api/` reads the request, builds a response dictionary using `G3JSON`, and writes the JSON string with the correct `Content-Type` header.

Example requests:

```
GET http://localhost:8801/rest/api/status.asp
GET http://localhost:8801/rest/api/echo.asp?name=AxonASP
```

696.5RESTful — Full RESTful API

Path: `./www/restful/` **URL:** `http://localhost:8801/restful/`

This example extends the REST example with a more complete RESTful design, including support for multiple HTTP verbs and resource-oriented endpoints.

Path	Description
`restful/default.asp`	Front-end page demonstrating API calls
`restful/api/`	API endpoint files organized by resource

How it works: Endpoint scripts branch on `Request.ServerVariables("REQUEST_METHOD")` to handle `GET`, `POST`, `PUT`, and `DELETE` verbs. Responses are JSON formatted using `G3JSON`.

696.6 Shared Patterns

All four examples follow these AxonASP Classic ASP conventions:

- `Option Explicit` is declared at the top of each file.
- Objects use `Set ... = Server.CreateObject(...)` syntax.
- JSON output uses `G3JSON` (`Server.CreateObject("G3JSON")`).
- CSS uses `../css/axonasp.css` from the shared stylesheet.
- No framework dependencies — pure Classic ASP and VBScript.

696.7 Remarks

- The example applications are for development and educational use only. They are not hardened for production deployment.
- To adapt an example as a starting point for a real project, copy the directory to a new location under `www/` and modify the files as needed.
- All four examples run without any database or external service. They can be exercised immediately after starting the AxonASP HTTP server.

697 Database Convert Tool

697.1 Overview

The Database Convert tool is a built-in AxonASP web application located at `./www/database-convert/` that provides a wizard-style interface for migrating Microsoft Access databases to other database formats supported by AxonASP. The tool reads the source Access database file, inspects its schema and data using ADODB, and writes the content to the selected target.

Access the tool at: `http://localhost:8801/database-convert/`

697.2 Supported Conversions

Source	Target	Status
Microsoft Access (.mdb / .accdb)	SQLite	Supported
Microsoft Access (.mdb / .accdb)	MySQL	Supported
Microsoft Access (.mdb / .accdb)	PostgreSQL	Supported
Microsoft Access (.mdb / .accdb)	Microsoft SQL Server	Supported

Source	Target	Status
Microsoft Access (.mdb / .accdb)	Oracle	Not supported

Note: Conversion to Oracle is not currently supported.

697.3 How It Works

The conversion runs as an ASP wizard with multiple steps:

1. **Upload step:** The user uploads a `.mdb` or `.accdb` Access database file.
2. **Target selection step:** The user selects the destination database type and provides connection details.
3. **Conversion step:** The worker script (`import_worker.asp`) opens the source database using ADODB with the ACE OLE DB or Jet OLE DB provider, reads each table's schema, creates equivalent tables in the target database, and inserts all rows in batches.

The tool uses: - `ADODB.Connection` to open both the source and the target database. - `G3FILES` for file handling and temporary storage of the uploaded database. - A target-specific connection string built from the connection parameters provided by the user.

697.4 Prerequisites

- **Windows only:** The source database connection requires the Microsoft Access Database Engine (ACE OLE DB 12.0) or the older Jet OLE DB 4.0 provider. These are Windows-only COM providers.
- The target database must be reachable from the machine running AxonASP.
- For SQLite: the `SQLITE_PATH` environment variable must point to the destination `.db` file.
- For MySQL, PostgreSQL, or SQL Server: valid host, port, database name, username, and password are required.

697.5 Identifier Quoting

The tool applies correct identifier quoting for each target dialect:

Target	Quoting style
SQLite	`[brackets]`
SQL Server	`[brackets]`
MySQL	`` `backticks` ``
PostgreSQL	`"double quotes"`

697.6 Type Mapping

ADO field types from Access are mapped to type-appropriate SQL column definitions in the target schema. Unsupported or unknown ADO types fall back to a generic text type.

697.7 Running the Tool

1. Start the AxonASP HTTP server.
2. Open a browser and navigate to `http://localhost:8801/database-convert/`.
3. Follow the wizard steps: upload the Access database, select the target, enter connection details, and start the conversion.

697.8 Remarks

- The tool is a development and administration utility. It is not intended for production-facing traffic. Restrict access to this path in production environments using web.config rewrite rules or network-level controls.
- Large databases with many rows may take time to convert. The conversion runs synchronously within the ASP request.
- The Access database file is temporarily stored on the server during conversion. It is removed after the conversion completes.
- Primary keys, indexes, and foreign key constraints from Access are not automatically recreated in the target schema. Only table structure and data are migrated.

698 Write New AxonASP Manual Pages

698.1 Overview

This document provides the standard procedure and strict rules for creating or updating AxonASP manual pages. It is designed to be read by both human technical writers and Large Language Models (LLMs) acting as documentation generators. The goal is consistent structure, precise API coverage, strict adherence to legacy standards, and compatibility with our UI tree-view rendering.

DO NOT write generic or assumed documentation. Be extremely specific about AxonASP's implementation.

698.2 Syntax Template

Every manual page must follow this exact structure:

```
# <Action-Oriented Title>

## Overview
```

```
## Syntax
## Parameters and Arguments
## Return Values
## Remarks
## Code Example
```

698.3Section Requirements

- **Action-Oriented Title (Required):** Include the object/member purpose directly in the title.
- **Overview (Required):** Short, precise explanation of what the page covers.
- **Syntax (Required):** Exact Classic ASP/VBScript syntax form.
- **Parameters and Arguments (Required):** Include data type, required/optional state, and specific expected values. No generic descriptions.
- **Return Values (Required):** Describe the exact expected output and failure behavior. Never use generic catch-all descriptions.
- **Remarks (Required):** Include compatibility notes, edge cases, performance details, and AxonASP-specific behaviors.
- **Code Example (Required):** Complete runnable Classic ASP example following the `llm-classic-asp-coding.md` guidelines.

698.4STRICT RULES & CONSTRAINTS (DO NOT IGNORE)

Whether you are a human writer or an LLM, you must strictly follow these rules:

1. **Format & Style:**
    - Follow the classic Microsoft Writing Style Guide (action-oriented titles, brief overviews, prerequisites, code examples, extra information on how the code works, and complete API references).
    - Use active voice, simple language, scannable lists, and **bold text** for emphasis.
    - **ABSOLUTELY NO EMOJIS.**
    - **NO MARKDOWN LINKS inside the content page.** Navigation links are strictly reserved for `www/manual/menu.md`.
    - Keep all text and examples in English (US).
2. **Branding:**
    - Use the “G3Pix AxonASP” name and branding.
    - **DO NOT** use “MSDN” or “Microsoft” names, trademarks, or logos anywhere in the text. Replicate their classic, rigorous aesthetic and structural style only.
3. **Instantiation Precision (ProgID):**
    - Never describe an object creation using aliases in the code snippet.
    - *Incorrect:* `Set obj = Server.CreateObject("G3CRYPTO and aliases")`

    - *Correct:* `Set obj = Server.CreateObject("G3CRYPTO")` (Always use the primary name only).
4. **Explicit Return Values:**
    - Do NOT use generic catch-all descriptions for return values.
    - *Incorrect:* "Returns a Variant result. Depending on the operation, this can be String, Boolean, Number, Array, Dictionary/object handle, or Empty."
    - *Correct:* You must document exactly what *that specific function* returns (e.g., "Returns a String containing the SHA-256 hash of the input", or "Returns a Boolean indicating whether the decryption was successful").
    - **DON'T BE GENERALIST. BE VERY SPECIFIC.**
5. **Menu Navigation (`www/manual/menu.md`):**
    - Always use markdown links for navigable entries with the exact format `[Label](folder/document.md)`.
    - Keep folder/group labels in `menu.md` as plain bullet items when they must have child items.
    - Menu link targets must be paths relative to `www/manual/md`.
6. **Summary Pages (`methods.md` and `properties.md`):**
    - When creating summary or index pages for a library (e.g., `/<library_name>/methods.md` or `/<library_name>/properties.md`), you MUST use markdown tables to list the members.
    - For `methods.md`, use the columns: `Method`, `Returns`, `Description` (e.g., see `/g3crypto/methods.md` style).
    - For `properties.md`, use the columns: `Property`, `Access`, `Type`, `Description`.

698.5 LLM Authoring Prompt

If you are using an LLM to generate or update documentation for AxonASP, provide the AI with the source code (e.g., the `lib_*.go` file) and use the following prompt to ensure strict compliance:

```
Act as an expert technical writer for the AxonASP project (developed by G
3Pix). We need to write or significantly improve the documentation for th
e provided AxonASP library/object. Your writing is focused on ASP Classic,
 GoLang and VBScript.

Your task is to write the documentation to be precise, professional, and
strictly compliant with our documentation standards.

**STRICT RULES & CONSTRAINTS (DO NOT IGNORE):**

1. **Format & Style:** Follow the Microsoft Writing Style Guide (action-o
```

```
riented titles, brief overviews, prerequisites, code examples, extra info
rmation on how the code works, and complete API references). Use active v
oice, simple language, scannable lists, and bold text for emphasis. ABSOL
UTELY NO EMOJIS. NO MARKDOWN LINKS inside the content page.
2. **Template:** Use exactly these headers: `# Title`, `## Overview`, `##
 Syntax`, `## Parameters and Arguments`, `## Return Values`, `## Remarks`,
 `## Code Example`. For summary pages (`methods.md` or `properties.md`),
use markdown tables. `methods.md` needs `Method | Returns | Description`.
 `properties.md` needs `Property | Access | Type | Description`.
3. **Branding:** Use the "G3Pix AxonASP" name. DO NOT use "MSDN" or "Micr
osoft" names or logos.
4. **Instantiation Precision (ProgID):** Never describe an object creatio
n using aliases. e.g., use `Set obj = Server.CreateObject("PRIMARY_NAME_O
NLY")`.
5. **Explicit Return Values:** Do NOT use generic catch-all descriptions.
 Document exactly what THAT SPECIFIC function returns (e.g., "Returns a S
tring containing..."). DON'T BE GENERALIST. BE VERY SPECIFIC.
6. **Code Example:** Provide a complete, runnable Classic ASP example in
English (US). Ensure the code is realistic, correct and demonstrates actu
al usage.
7. When creating code you can also check llm-classic-asp-coding.md in the
 manual, to ensure the ASP Classic code is following the guidelines.

Here is the source code / library details to document:
[INSERT SOURCE CODE OR DETAILS HERE]
```

698.6 Example Code Guideline

```asp
<%
Dim crypto
' Correct primary instantiation
Set crypto = Server.CreateObject("G3CRYPTO")

Dim hashValue
' Precise documentation of the return value is expected in the text above
 this example
hashValue = crypto.MD5("my_secret_string")

If Len(hashValue) > 0 Then
    Response.Write "Hash generated successfully."
End If

Set crypto = Nothing
%>
```

699 Program Classic ASP with LLM/AI agents in AxonASP

699.1 Overview

Use this guide when asking LLM/AI agents to generate or refactor Classic ASP VBScript code for AxonASP. The objective is strict compatibility, predictable runtime behavior, and safe code generation aligned with project standards.

699.2 Syntax

```
Task for AI agent:
- Target runtime: Classic ASP VBScript 5.8
- Enforce: block If...Then...End If only
- Enforce: Set for object assignment
- Enforce: & for string concatenation
- Enforce: Option Explicit
- Enforce: no inline Dim initialization
```

699.3 Parameters and Arguments

- Runtime target (Required): Classic ASP with VBScript 5.8 semantics.
- Directive format (Required): Keep ‹%@ … %› on the first line, compact, and fully closed.
- Control flow policy (Required): Use block If…Then…End If only. Do not use single-line If statements.
- Variable policy (Required): Use Option Explicit and separate declaration from assignment.
- Object policy (Required): Use Set for object references and Set obj = Nothing for cleanup.
- Call syntax policy (Required): For Sub/method calls without return, use no parentheses or use Call with parentheses.
- Function call policy (Required): Use parentheses when assigning function return values.
- Error policy (Required): Use On Error Resume Next, check Err.Number, and reset with On Error GoTo 0.
- Boolean policy (Required): Avoid short-circuit assumptions for And and Or; nest conditions where safety is required.
- Concatenation policy (Required): Use & for strings. Do not use + for string concatenation.
- Object creation policy (Required): Use exact ProgID values for Server.CreateObject.

699.4 Return Values

Using these constraints in prompts produces code that is more likely to execute without parser or runtime regressions in Classic ASP and AxonASP.

699.5 Remarks

- Treat every generated snippet as compatibility-sensitive code.
- Ask the LLM to avoid modern language patterns such as try/catch, typed declarations, or inline initialization.
- Require explicit object lifecycle cleanup in long-running scripts.
- Require defensive condition nesting for recordset and object access.
- Validate generated ProgIDs against the AxonASP manual before execution.

699.6 Agent Prompting Example

When asking an AI agent to generate Classic ASP code for AxonASP, include the following checklist in your prompt to ensure compliance with the coding standards outlined above:

```
STRICT CLASSIC ASP & VBSCRIPT CODING GUIDELINES

Overview: Classic ASP (Active Server Pages) powered by VBScript 5.8 has a
 highly specific syntax that predates modern programming paradigms. To en
sure maximum compatibility, performance, and stability, all ASP code must
 strictly adhere to the following Microsoft IIS legacy standards.

1. ASP Directives and Delimiters
Modern parsers might be forgiving, but strict Classic ASP requires exact
formatting for page directives.
* No Line Breaks in Directives: The processing directive `<%@ ... %>` mus
t exist entirely on the very first line of the file. You cannot break it
across multiple lines.
* No Extra Spacing: Keep directives compact. Do not add unnecessary space
s around the equal signs.
    * Correct: `<%@Language="VBSCRIPT" CodePage="65001"%>`
    * Incorrect: `<%@ Language = "VBSCRIPT" %>`
* Tag Closure:Never leave `<%` or `%>` tags unclosed or mismatched.

2. Control Structures (The "If-Then" Rule)
VBScript control structures are rigid. Modern shortcuts do not apply.
* Mandatory Block `If...Then...End If`: You must never use single-line `I
f...Then` statements (e.g., `If x = 1 Then y = 2`). While legacy VBScript
 technically allowed this, it is notoriously prone to parsing errors and
scope bleeding in complex ASP pages. You MUST always use block syntax and
 explicitly close the statement with `End If`.
    * Correct:
        If x = 1 Then
```

```
            y = 2
        End If
```
    * Incorrect: `If x = 1 Then y = 2`
* Loop Closures: `For` loops must end with `Next` (optionally `Next variableName`), `Do While` must end with `Loop`, and `While` must end with `Wend`.

3. Variable Declaration & Initialization
VBScript lacks modern variable declaration features.
* Mandatory `Option Explicit`: Every single VBScript page or included file should ideally enforce `Option Explicit` to prevent undeclared variables.
* No Inline Initialization: You cannot declare and assign a variable on the same line.
    * Correct:
```
        Dim myVar
        myVar = "Hello"
```
    * Incorrect: `Dim myVar = "Hello"`
* Everything is a Variant: There is no strong typing in VBScript. Do not attempt to type variables (e.g., `Dim x As Integer` will throw a syntax error).

4. Object Assignment (`Set` vs. `Let`)
This is the most common pitfall for modern developers writing VBScript.
* The `Set` Keyword: Whenever you are assigning an Object (like a Recordset, Connection, FSO, or Dictionary) to a variable, you must use the `Set` keyword.
    * Correct: `Set rs = Server.CreateObject("ADODB.Recordset")`
    * Incorrect: `rs = Server.CreateObject("ADODB.Recordset")`
* The `Nothing` Keyword: To destroy an object and free memory, you must explicitly set it to `Nothing`.
    * Correct: `Set rs = Nothing`

5. Method and Function Calling (Parentheses Rules)
VBScript's rules for parentheses are highly idiosyncratic compared to modern languages.
* Calling Subs or Methods (No Return Value): If you are calling a Sub or a method without expecting a return value, do not use parentheses, OR use the `Call` keyword with parentheses.
    * Correct: `Response.Write "Hello World"`
    * Correct: `Call Response.Write("Hello World")`
    * Incorrect: `Response.Write("Hello World")` (This passes the argument ByVal and can cause syntax errors with multiple arguments).
* Calling Functions (Expecting a Return Value): If you are assigning the result to a variable, you must use parentheses.
    * Correct: `myLen = Len("Hello")`

6. Major Quirks vs. Modern Languages
* No Short-Circuit Logic: In an `If A And B Then` statement, VBScript eva

luates both `A` and `B`, even if `A` is false. This will cause crashes if `B` relies on `A` being true (e.g., `If Not rs.EOF And rs("Name") = "John" Then` will crash if the recordset is at EOF). You must nest your `If` statements.
* Error Handling: There is no `try...catch...finally` block. You must use inline error handling.
    * Enable: `On Error Resume Next`
    * Check: `If Err.Number <> 0 Then ...`
    * Disable/Reset: `On Error GoTo 0`
* String Concatenation: Always use the ampersand (`&`) to concatenate strings, never the plus sign (`+`), as `+` can inadvertently cause mathematical addition if both variants evaluate to numbers.
* Equality Operator: A single equals sign (`=`) acts as both the assignment operator AND the equality comparison operator. There is no `==` or `===`.
    * Assignment: `x = 5`
    * Comparison: `If x = 5 Then`
* Line Continuation: Statements cannot naturally break across lines. You must use the underscore character (`_`) preceded by a space to span a single logical line of code across multiple physical lines.

7. Server-Side Object Creation
When using `Server.CreateObject`, you must use the exact ProgID string as defined by the library or component. The server will return a native object handle that is compatible with VBScript's late binding. Always check the documentation for the correct ProgID and ensure that the corresponding library is properly registered on the server.

8. Use of the MCP Server and AxonASP Libraries
When creating objects with `Server.CreateObject`, prefer using the ProgIDs provided by the AxonASP manual for maximum compatibility and performance. These ProgIDs are designed to leverage the native implementations in AxonASP, which are optimized for the Classic ASP environment.
If you have access to the AxonASP MCP server, you should query it for information on code formatting, as well as available server functions and properties. If the MCP server is unavailable, refer to the manual located in ./www/manual/md or online at https://github.com/guimaraeslucas/axonasp/tree/main/www/manual/md/authoring/llm-classic-asp-coding.md.
The following objects are available natively. To ensure maximum performance and efficiency, always prioritize using these built-in objects over writing custom ASP scripts for the same functionality: G3MD, G3CRYPTO, G3Axon, G3JSON, G3DB, G3HTTP, G3MAIL, G3Image, G3FILES, G3Template, G3Zip, G3ZLIB, G3TAR, G3ZSTD, G3FC, WScript.Shell, ADOX.Catalog, MSWC.AdRotator, MSWC.BrowserType, MSWC.NextLink, MSWC.ContentRotator, MSWC.Counters, MSWC.PageCounter, MSWC.Tools, MSWC.MyInfo, MSWC.PermissionChecker, MSXML2.ServerXMLHTTP, MSXML2.XMLHTTP, Microsoft.XMLHTTP, MSXML2.DOMDocument, Microsoft.XMLDOM, G3PDF, G3FileUploader and upload compatibility aliases, Scripting.FileSystemObject, Scripting.Dictionary, ADODB.Stream, ADODB.Connection, ADODBOLE.Connection, ADODB.Recordset, ADODB.Command, VBScript.RegExp and RegExp.

699.7 Code Example

```
<%@Language="VBSCRIPT"%>
<%
Option Explicit

Dim conn
Dim rs
Dim sql

Set conn = Server.CreateObject("ADODB.Connection")
conn.ConnectionString = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" &
 Server.MapPath("./data.mdb")
conn.Open

sql = "SELECT name FROM users"
Set rs = Server.CreateObject("ADODB.Recordset")
rs.Open sql, conn

If Not rs.EOF Then
    Response.Write rs("name")
End If

rs.Close
conn.Close
Set rs = Nothing
Set conn = Nothing
%>
```

700 Legal Disclaimer and License Information

700.1 Product Information

G3pix AxonASP Copyright (C) 2026 G3pix Ltda. All rights reserved.

- **Developer:** Lucas Guimaraes - G3pix Ltda
- **Contact:** https://g3pix.com.br/
- **Project URL:** https://g3pix.com.br/axonasp

700.21. Software License

This Source Code Form is subject to the terms of the **Mozilla Public License, v. 2.0**.

- **Attribution Notice:** If this software is used in other projects, the name “G3pix AxonASP” must be cited in the documentation or About section.

- **Contribution Policy:** Modifications to the core source code of G3pix AxonASP must be made available under these same MPL 2.0 license terms.

```
Mozilla Public License Version 2.0
==================================

1. Definitions
--------------

1.1. "Contributor"
    means each individual or legal entity that creates, contributes to
    the creation of, or owns Covered Software.

1.2. "Contributor Version"
    means the combination of the Contributions of others (if any) used
    by a Contributor and that particular Contributor's Contribution.

1.3. "Contribution"
    means Covered Software of a particular Contributor.

1.4. "Covered Software"
    means Source Code Form to which the initial Contributor has attached
    the notice in Exhibit A, the Executable Form of such Source Code
    Form, and Modifications of such Source Code Form, in each case
    including portions thereof.

1.5. "Incompatible With Secondary Licenses"
    means

    (a) that the initial Contributor has attached the notice described
        in Exhibit B to the Covered Software; or

    (b) that the Covered Software was made available under the terms of
        version 1.1 or earlier of the License, but not also under the
        terms of a Secondary License.

1.6. "Executable Form"
    means any form of the work other than Source Code Form.

1.7. "Larger Work"
    means a work that combines Covered Software with other material, in
    a separate file or files, that is not Covered Software.

1.8. "License"
    means this document.

1.9. "Licensable"
    means having the right to grant, to the maximum extent possible,
```

whether at the time of the initial grant or subsequently, any and all of the rights conveyed by this License.

1.10. "Modifications"
means any of the following:

(a) any file in Source Code Form that results from an addition to, deletion from, or modification of the contents of Covered Software; or

(b) any new file in Source Code Form that contains any Covered Software.

1.11. "Patent Claims" of a Contributor
means any patent claim(s), including without limitation, method, process, and apparatus claims, in any patent Licensable by such Contributor that would be infringed, but for the grant of the License, by the making, using, selling, offering for sale, having made, import, or transfer of either its Contributions or its Contributor Version.

1.12. "Secondary License"
means either the GNU General Public License, Version 2.0, the GNU Lesser General Public License, Version 2.1, the GNU Affero General Public License, Version 3.0, or any later versions of those licenses.

1.13. "Source Code Form"
means the form of the work preferred for making modifications.

1.14. "You" (or "Your")
means an individual or a legal entity exercising rights under this License. For legal entities, "You" includes any entity that controls, is controlled by, or is under common control with You. For purposes of this definition, "control" means (a) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (b) ownership of more than fifty percent (50%) of the outstanding shares or beneficial ownership of such entity.

2. License Grants and Conditions

2.1. Grants

Each Contributor hereby grants You a world-wide, royalty-free, non-exclusive license:

(a) under intellectual property rights (other than patent or trademark) Licensable by such Contributor to use, reproduce, make available, modify, display, perform, distribute, and otherwise exploit its Contributions, either on an unmodified basis, with Modifications, or as part of a Larger Work; and

(b) under Patent Claims of such Contributor to make, use, sell, offer for sale, have made, import, and otherwise transfer either its Contributions or its Contributor Version.

2.2. Effective Date

The licenses granted in Section 2.1 with respect to any Contribution become effective for each Contribution on the date the Contributor first distributes such Contribution.

2.3. Limitations on Grant Scope

The licenses granted in this Section 2 are the only rights granted under this License. No additional rights or licenses will be implied from the distribution or licensing of Covered Software under this License. Notwithstanding Section 2.1(b) above, no patent license is granted by a Contributor:

(a) for any code that a Contributor has removed from Covered Software; or

(b) for infringements caused by: (i) Your and any other third party's modifications of Covered Software, or (ii) the combination of its Contributions with other software (except as part of its Contributor Version); or

(c) under Patent Claims infringed by Covered Software in the absence of its Contributions.

This License does not grant any rights in the trademarks, service marks, or logos of any Contributor (except as may be necessary to comply with the notice requirements in Section 3.4).

2.4. Subsequent Licenses

No Contributor makes additional grants as a result of Your choice to distribute the Covered Software under a subsequent version of this License (see Section 10.2) or under the terms of a Secondary License (if permitted under the terms of Section 3.3).

2.5. Representation

Each Contributor represents that the Contributor believes its Contributions are its original creation(s) or it has sufficient rights to grant the rights to its Contributions conveyed by this License.

2.6. Fair Use

This License is not intended to limit any rights You have under applicable copyright doctrines of fair use, fair dealing, or other equivalents.

2.7. Conditions

Sections 3.1, 3.2, 3.3, and 3.4 are conditions of the licenses granted in Section 2.1.

3. Responsibilities

3.1. Distribution of Source Form

All distribution of Covered Software in Source Code Form, including any Modifications that You create or to which You contribute, must be under the terms of this License. You must inform recipients that the Source Code Form of the Covered Software is governed by the terms of this License, and how they can obtain a copy of this License. You may not attempt to alter or restrict the recipients' rights in the Source Code Form.

3.2. Distribution of Executable Form

If You distribute Covered Software in Executable Form then:

(a) such Covered Software must also be made available in Source Code Form, as described in Section 3.1, and You must inform recipients of the Executable Form how they can obtain a copy of such Source Code Form by reasonable means in a timely manner, at a charge no more than the cost of distribution to the recipient; and

(b) You may distribute such Executable Form under the terms of this License, or sublicense it under different terms, provided that the license for the Executable Form does not attempt to limit or alter the recipients' rights in the Source Code Form under this License.

3.3. Distribution of a Larger Work

You may create and distribute a Larger Work under terms of Your choice,

provided that You also comply with the requirements of this License for the Covered Software. If the Larger Work is a combination of Covered Software with a work governed by one or more Secondary Licenses, and the Covered Software is not Incompatible With Secondary Licenses, this License permits You to additionally distribute such Covered Software under the terms of such Secondary License(s), so that the recipient of the Larger Work may, at their option, further distribute the Covered Software under the terms of either this License or such Secondary License(s).

3.4. Notices

You may not remove or alter the substance of any license notices (including copyright notices, patent notices, disclaimers of warranty, or limitations of liability) contained within the Source Code Form of the Covered Software, except that You may alter any license notices to the extent required to remedy known factual inaccuracies.

3.5. Application of Additional Terms

You may choose to offer, and to charge a fee for, warranty, support, indemnity or liability obligations to one or more recipients of Covered Software. However, You may do so only on Your own behalf, and not on behalf of any Contributor. You must make it absolutely clear that any such warranty, support, indemnity, or liability obligation is offered by You alone, and You hereby agree to indemnify every Contributor for any liability incurred by such Contributor as a result of warranty, support, indemnity or liability terms You offer. You may include additional disclaimers of warranty and limitations of liability specific to any jurisdiction.

4. Inability to Comply Due to Statute or Regulation

If it is impossible for You to comply with any of the terms of this License with respect to some or all of the Covered Software due to statute, judicial order, or regulation then You must: (a) comply with the terms of this License to the maximum extent possible; and (b) describe the limitations and the code they affect. Such description must be placed in a text file included with all distributions of the Covered Software under this License. Except to the extent prohibited by statute or regulation, such description must be sufficiently detailed for a recipient of ordinary skill to be able to understand it.

5. Termination

5.1. The rights granted under this License will terminate automatically

if You fail to comply with any of its terms. However, if You become compliant, then the rights granted under this License from a particular Contributor are reinstated (a) provisionally, unless and until such Contributor explicitly and finally terminates Your grants, and (b) on an ongoing basis, if such Contributor fails to notify You of the non-compliance by some reasonable means prior to 60 days after You have come back into compliance. Moreover, Your grants from a particular Contributor are reinstated on an ongoing basis if such Contributor notifies You of the non-compliance by some reasonable means, this is the first time You have received notice of non-compliance with this License from such Contributor, and You become compliant prior to 30 days after Your receipt of the notice.

5.2. If You initiate litigation against any entity by asserting a patent infringement claim (excluding declaratory judgment actions, counter-claims, and cross-claims) alleging that a Contributor Version directly or indirectly infringes any patent, then the rights granted to You by any and all Contributors for the Covered Software under Section 2.1 of this License shall terminate.

5.3. In the event of termination under Sections 5.1 or 5.2 above, all end user license agreements (excluding distributors and resellers) which have been validly granted by You or Your distributors under this License prior to termination shall survive termination.

```
************************************************************************
*                                                                      *
*  6. Disclaimer of Warranty                                           *
*  -------------------------                                           *
*                                                                      *
*  Covered Software is provided under this License on an "as is"       *
*  basis, without warranty of any kind, either expressed, implied, or  *
*  statutory, including, without limitation, warranties that the       *
*  Covered Software is free of defects, merchantable, fit for a        *
*  particular purpose or non-infringing. The entire risk as to the     *
*  quality and performance of the Covered Software is with You.        *
*  Should any Covered Software prove defective in any respect, You     *
*  (not any Contributor) assume the cost of any necessary servicing,   *
*  repair, or correction. This disclaimer of warranty constitutes an   *
*  essential part of this License. No use of any Covered Software is   *
*  authorized under this License except under this disclaimer.         *
*                                                                      *
************************************************************************

************************************************************************
*                                                                      *
*  7. Limitation of Liability                                          *
*  --------------------------                                          *
*                                                                      *
```

* Under no circumstances and under no legal theory, whether tort   *
* (including negligence), contract, or otherwise, shall any   *
* Contributor, or anyone who distributes Covered Software as   *
* permitted above, be liable to You for any direct, indirect,   *
* special, incidental, or consequential damages of any character   *
* including, without limitation, damages for lost profits, loss of   *
* goodwill, work stoppage, computer failure or malfunction, or any   *
* and all other commercial damages or losses, even if such party   *
* shall have been informed of the possibility of such damages. This   *
* limitation of liability shall not apply to liability for death or   *
* personal injury resulting from such party's negligence to the   *
* extent applicable law prohibits such limitation. Some   *
* jurisdictions do not allow the exclusion or limitation of   *
* incidental or consequential damages, so this exclusion and   *
* limitation may not apply to You.   *
*   *
****

8. Litigation

Any litigation relating to this License may be brought only in the courts of a jurisdiction where the defendant maintains its principal place of business and such litigation shall be governed by laws of that jurisdiction, without reference to its conflict-of-law provisions. Nothing in this Section shall prevent a party's ability to bring cross-claims or counter-claims.

9. Miscellaneous

This License represents the complete agreement concerning the subject matter hereof. If any provision of this License is held to be unenforceable, such provision shall be reformed only to the extent necessary to make it enforceable. Any law or regulation which provides that the language of a contract shall be construed against the drafter shall not be used to construe this License against a Contributor.

10. Versions of the License

10.1. New Versions

Mozilla Foundation is the license steward. Except as provided in Section 10.3, no one other than the license steward has the right to modify or publish new versions of this License. Each version will be given a distinguishing version number.

10.2. Effect of New Versions

You may distribute the Covered Software under the terms of the version of the License under which You originally received the Covered Software, or under the terms of any subsequent version published by the license steward.

10.3. Modified Versions

If you create software not governed by this License, and you want to create a new license for such software, you may create and use a modified version of this License if you rename the license and remove any references to the name of the license steward (except to note that such modified license differs from this License).

10.4. Distributing Source Code Form that is Incompatible With Secondary Licenses

If You choose to distribute Source Code Form that is Incompatible With Secondary Licenses under the terms of this version of the License, the notice described in Exhibit B of this License must be attached.

Exhibit A - Source Code Form License Notice

  This Source Code Form is subject to the terms of the Mozilla Public License, v. 2.0. If a copy of the MPL was not distributed with this file, You can obtain one at http://mozilla.org/MPL/2.0/.

If it is not possible or desirable to put the notice in a particular file, then You may include the notice in a location (such as a LICENSE file in a relevant directory) where a recipient would be likely to look for such a notice.

You may add additional accurate notices of copyright ownership.

Exhibit B - "Incompatible With Secondary Licenses" Notice

  This Source Code Form is "Incompatible With Secondary Licenses", as defined by the Mozilla Public License, v. 2.0.

700.32. G3pix Trademarks and Logo Usage

G3pix is a registered trademark of G3pix Ltda (2026). All rights are reserved, and unauthorized use of the trademark alone is strictly prohibited and requires previous authorization.

Logo Permitted Use: The use of the complete AxonASP logo featuring the G3pix inscription is permitted for advertising, educational, and general informational materials. This permission is granted provided that the materials do not contain illegal or fraudulent content, and are not used for purposes foreign to the intended functionality and nature of the software.

700.43. Third-Party Trademarks and Affiliations

AxonASP is an independent software project developed by G3pix Ltda and is **not affiliated with, endorsed by, or sponsored by Microsoft Corporation** in any way.

The names “Microsoft,” “Active Server Pages,” “ASP,” “VBScript,” “Windows,” “Office,” “Access,” “ActiveX,” as well as any related names, marks, emblems, and images relative to ASP, are registered trademarks of Microsoft Corporation. The use of these trademarks within this project is purely for descriptive, identification, and reference purposes to indicate technical compatibility, and does not imply any association with the trademark holder.

700.54. Disclaimer of Warranty

Covered Software is provided under this License on an “as is” basis, without warranty of any kind, either expressed, implied, or statutory, including, without limitation, warranties that the Covered Software is free of defects, merchantable, fit for a particular purpose or non-infringing. The entire risk as to the quality and performance of the Covered Software is with You. Should any Covered Software prove defective in any respect, You (not any Contributor) assume the cost of any necessary servicing, repair, or correction. This disclaimer of warranty constitutes an essential part of this License. No use of any Covered Software is authorized under this License except under this disclaimer.

700.65. VBScript Package License

The following license applies to the vbscript package:

```
THIRD PARTY ATTRIBUTION / ORIGINAL SOURCE

 This code was adapted from https://github.com/kmvi/vbscript-parser/
 ensuring compatibility with VBScript language specifications.

 Original Copyright (c) kmvi (and/or original authors).
 Licensed under the BSD 3-Clause License.

 Redistribution and use in source and binary forms, with or without
 modification, are permitted provided that the following conditions are m
et:
```

```
 1. Redistributions of source code must retain the above copyright notice,
 this list of conditions and the following disclaimer.

 2. Redistributions in binary form must reproduce the above copyright not
ice,
 this list of conditions and the following disclaimer in the documentatio
n
 and/or other materials provided with the distribution.

 3. Neither the name of the copyright holder nor the names of its
 contributors may be used to endorse or promote products derived from
 this software without specific prior written permission.

 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS
IS"
 AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, TH
E
 IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPO
SE
 ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS B
E
 LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
 CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
 SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
 INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
 CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF T
HE
 POSSIBILITY OF SUCH DAMAGE.
```

700.76. AxonASP Dependencies and Licenses

These third-party libraries are used in the AxonASP project, along with their detected licenses:

```
codeberg.org/go-pdf/fpdf (v0.11.1) - MIT
filippo.io/edwards25519 (v1.2.0) - BSD-3-Clause
github.com/bahlo/generic-list-go (v0.2.0) - BSD-3-Clause
github.com/buger/jsonparser (v1.1.2) - MIT
github.com/cespare/xxhash/v2 (v2.3.0) - MIT
github.com/denisenkom/go-mssqldb (v0.12.3) - BSD-3-Clause
github.com/dustin/go-humanize (v1.0.1) - MIT
github.com/fogleman/gg (v1.3.0) - MIT
github.com/fsnotify/fsnotify (v1.9.0) - BSD-3-Clause
github.com/fxamacker/cbor/v2 (v2.9.1) - MIT
github.com/gdamore/encoding (v1.0.1) - Apache-2.0
github.com/gdamore/tcell/v2 (v2.13.8) - Apache-2.0
github.com/go-ole/go-ole (v1.3.0) - MIT
github.com/go-sql-driver/mysql (v1.9.3) - MPL-2.0
```

github.com/go-viper/mapstructure/v2 (v2.5.0) - MIT
github.com/golang-sql/civil (v0.0.0-20220223132316-b832511892a9) - Apache
-2.0
github.com/golang-sql/sqlexp (v0.1.0) - BSD-3-Clause AND LicenseRef-scanc
ode-google-patent-license-golang
github.com/golang/freetype (v0.0.0-20170609003504-e2365dfdc4a0) - FTL OR
GPL-2.0-or-later
github.com/goodsign/monday (v1.0.2) - BSD-2-Clause
github.com/google/jsonschema-go (v0.4.2) - MIT
github.com/google/uuid (v1.6.0) - BSD-3-Clause
github.com/hashicorp/golang-lru/v2 (v2.0.7) - MPL-2.0
github.com/invopop/jsonschema (v0.13.0) - MIT
github.com/joho/godotenv (v1.5.1) - MIT
github.com/kardianos/service (v1.2.4) - Zlib
github.com/klauspost/compress (v1.18.5) - Apache-2.0 AND BSD-3-Clause AND
 MIT
github.com/klauspost/cpuid/v2 (v2.3.0) - MIT
github.com/klauspost/reedsolomon (v1.13.3) - MIT
github.com/kylelemons/godebug (v1.1.0) - Apache-2.0
github.com/lib/pq (v1.12.3) - MIT
github.com/lucasb-eyer/go-colorful (v1.4.0) - MIT
github.com/mailru/easyjson (v0.9.2) - MIT
github.com/mark3labs/mcp-go (v0.47.1) - MIT
github.com/ncruces/go-strftime (v1.0.0) - MIT
github.com/pelletier/go-toml/ (v2.3.0) - MIT
github.com/remyoudompheng/bigfft (v0.0.0-20230129092748-24d4a6f8daec) - B
SD-3-Clause
github.com/ricochet2200/go-disk-usage/du (v0.0.0-20210707232629-ac9918953
285) - The Unlicense
github.com/rivo/tview (v0.42.0) - MIT
github.com/rivo/uniseg (v0.4.7) - MIT
github.com/sagikazarmark/locafero (v0.12.0) - MIT
github.com/sahilm/fuzzy (v0.1.1) - MIT
github.com/sijms/go-ora/v2 (v2.9.0) - MIT
github.com/spf13/afero (v1.15.0) - Apache-2.0
github.com/spf13/cast (v1.10.0) - MIT
github.com/spf13/pflag (v1.0.10) - BSD-3-Clause
github.com/spf13/viper (v1.21.0) - MIT
github.com/subosito/gotenv (v1.6.0) - MIT
github.com/wk8/go-ordered-map/v2 (v2.1.8) - Apache-2.0
github.com/x448/float16 (v0.8.4) - MIT
github.com/yosida95/uritemplate/v3 (v3.0.2) - BSD-3-Clause
github.com/yuin/goldmark (v1.8.2) - MIT
go.yaml.in/yaml/v3 (v3.0.4) - Apache-2.0 AND MIT
gopkg.in/alexcesaro/quotedprintable.v3 (v3.0.0-20150716171945-2caba252f4d
c) - MIT
gopkg.in/check.v1 (v1.0.0-20201130134442-10cb98267c6c) - BSD-2-Clause
gopkg.in/gomail.v2 (v2.0.0-20160411212932-81ebce5c23df) - MIT
gopkg.in/yaml.v3 (v3.0.1) - Apache-2.0
modernc.org/libc (v1.72.0) - BSD-3-Clause

modernc.org/mathutil (v1.7.1) - BSD-3-Clause
modernc.org/memory (v1.11.0) - BSD-3-Clause
modernc.org/sqlite (v1.48.2) - BSD-3-Clause