GoldMine API Guide

Release 2019.1

GoldMine

Copyright Notice

This document is provided strictly as a guide. No guarantees can be provided or expected. This document contains the confidential information and/or proprietary property of Ivanti, Inc. and its affiliates (referred to collectively as "Ivanti"), and may not be disclosed or copied without prior written consent of Ivanti.

Ivanti retains the right to make changes to this document or related product specifications and descriptions, at any time, without notice. Ivanti makes no warranty for the use of this document and assumes no responsibility for any errors that can appear in the document nor does it make a commitment to update the information contained herein. For the most current product information, please visit www.ivanti.com.

Copyright © 2019, Ivanti. All rights reserved.

Ivanti and its logos are registered trademarks or trademarks of Ivanti, Inc. and its affiliates in the United States and/or other countries. Other brands and names may be claimed as the property of others.

Protected by patents, see http://www.ivanti.com/patents

Updated: 2019

Contents

Contents 3
Introduction to Integrating with GoldMine 34
Introduction 34
Methods of Integrating with GoldMine 34
Integrating via Dynamic Data Exchange 35
Integrating via GMXS32.DLL 35
Integrating via the GoldMine XML API (GMXMLAPI.DLL) 35
Interacting with GoldMine via the GoldMine COM Server 35
Integrating via GoldMine Plug-ins 35
Integrating via a Database Engine 36
Comparing Integration Methods 36
Resources and Support 37
Technology Partner Program 37
Open Developer Community 37
Technology Partner Program 38
Integration Tools 38
Working with Dynamic Data Exchange (DDE) 39
Overview 39
Using DDE in GoldMine 39
Merging Data into a Document 39
Updating Database Information 40
Querying for Data 40
Identifying Telephone Numbers Automatically 40
Linking Contact Records to an Accounting Application 40
Inserting Incoming E-mail 40
Linking GoldMine to MS Word for Windows 41
Entering Application, Topic, and Item Names 41
DDE Parameters, Functions, Expressions, Macros 41
Establishing a DDE Conversation 42
To Initiate a DDE Conversation 42
To Request Data 43
Working with DDE Functions 43
Accessing Data Files 43
Adding an Empty Record 44
Parameters 44
Return Value 44
Example 44
Closing an Opened File 45
Parameters 45
Return Value 45
Example 45
Deleting the Current Record 45
Parameters 45
Example 45
Creating a Subset of Records 45
Parameters 46

Example 46
Checking for an Xbase or SQL Table 47
Parameters 47
Return Values 47
Moving to a Specified Record 47
Parameters 47
Return Value 49
Example 49
Opening a Data File 50
Parameters 51
Return Value 51
Example 51
Limiting GoldMine Search Range 51
Parameters 51
Example 52
Reading a Field Value 52
Parameters 52
Return Value 52
Checking the Current Record Number or Record ID 52
Parameters 52
Return Value 53
Example 53
Changing a Field Value 53
Parameters 53
Return Value 54
Example 54
Performing a Sequential Search 54
Parameters 54
Return Value 55
Search Return Values 55
Example 55
Unlocking a Record 56
Parameters 56
Return Value 56
Example 56
Accessing Contact Records 56
Linking GoldMine Fields with an External Application 56
Parameters 57
Valid RecordObj Functions 57
Return Value 60
Example 60
Accessing Specialized DDE Functions 61
Retrieving Login Credentials for Use with the GMXS32.DLL 61
Example 61
Retrieving the RecID of the Current Opportunity 62
Return Value 62
Example 62
Completing a Calendar Activity 62
Parameters 62

Return Value 63
Example 63
Displaying the Contact Record of an Incoming Caller 63
Parameters 64
CallerID Parameters 64
Return Values 64
CallerID Return Values 64
Example 65
Running a Counter 65
Parameters 65
Return Value 65
Example 65
Returning GoldMine Record Data 65
Record Selection 66
Parameters 66
Return Value 66
Parameters 67
Example 1 67
Example 2 67
Return Packet 67
Performance 68
Example 3 68
Example 5 69
Processing a Web Import Instruction File 69
Reading an Xbase Expression Without Opening a File 69
Parameters 69
Return Value 69
Example 69
Adding Merge Fields to a Form 70
Parameters 70
Example 70
Deleting Fields from a Form 71
Parameters 71
Return Value 72
Example 72
Closing a Form Profile 72
Parameters 72
Example 72
Creating an Xbase File with Registered Fields 72
Parameters 72
Examples of WhichRec Parameter 73
Return Value 73
Example 73
Returning a Field Name for an Expression 73
Parameters 73
Returning a Value for Unattached Fields 73
Example 73
Counting the Number of Exported Records 73
Parameters 74

Return Value 74
Example 74
Creating a History Record 74
Parameters 74
InsHistory Valid Values (2nd parameter) 74
Return Value 75
Example 75
Creating or Updating a Document Link 76
Parameters 76
Return Value 76
Example 77
Displaying a Message Dialog Box 77
Parameters 77
Return Value 78
Example 78
Adding a Merge Form 79
Parameters 79
Return Value 80
Example 80
Creating a Group 81
Parameters 81
Return Value 81
Example 81
Adding a Group Member 82
Parameters 82
Example 82
Creating a Macro 82
Parameters 83
Identifying a Macro by Number 83
Identifying a Macro by File Name 83
Return Value 83
Example 83
To Play a Macro from the Command Line 84
Creating and Sending a Pager Message 84
Return Value 85
SendPage Return Values 85
Example 85
Displaying a Message in the GoldMine Status Bar 85
Parameters 85
Example 86
Converting TLog Timestamps 86
Parameter 86
Return Values 86
Example 1 86
Example 2 86
DDE Macros 86
DDE Macros for Merge Forms 93
&PARAM2 Parameters 94
&PARAM4 Parameters 94

&PARAM5 Parameters 95
DDE Macros for the GoldMine License 95
Using GMXS32.DLL for Database Access and Sync Log Updates 97
Overview 97
Passing Multiple Parameters to a Function 97
Comparing Low Level/DDE Methodology to Business Logic Methodology 98
Method 1: Updating a Contact Record using the low level functions or DDE 98
Method 2: Updating a Contact Record using the Business Logic 98
Loading GMXS32.DLL and Logging In 98
For GoldMine Version 6.7 or Lower 99
Setting the SQL Database Login Name and PasswordGoldMine 6.7 or lower only) 99
Syntax 99
Parameters 99
Return Values 99
Example 99
Loading an API Session (GoldMine 7.0 or higher) 100
Parameters 100
Return Values 100
Notes 101
Example 101
Loading a BDE Session (GoldMine 6.7 or lower) 101
Syntax 101
Parameters 101
Return Values 101
Notes 102
Example 102
Logging in a User 102
Syntax 102
Parameters 103
Return Values 103
Example 103
Closing an API Session (GoldMine 7.0 or higher) 103
Syntax 103
Return Values 103
Notes 104
Example 104
Closing a BDE Session (GoldMine 6.7 or lower) 104
Syntax 104
Return Values 104
Notes 104
Example 104
Logging in Multiple Users through the API 104
Logging In 105
Syntax 105
Parameters 105
Return Values 105
Logging Out 106
Syntax 106
Parameters 106

Returns ... 106
Switching Between Login Sessions ... 106
Syntax ... 106
Parameters ... 106
Returns ... 106
Special Consideration for Multi-Threaded Applications ... 106
Syntax ... 107
Working with Business Logic Functions using the Name/Value Pair Method ... 107
Notes ... 107
Creating an NV Container ... 107
Syntax ... 107
Example ... 108
Return Value ... 108
Creating an NV Container with Copied Values ... 108
Syntax ... 108
Example ... 108
Return Value ... 108
Syntax ... 108
Parameters ... 108
Example ... 108
Return Value ... 108
Deleting an NV Container ... 109
Syntax ... 109
Example ... 109
Return Value ... 109
Syntax ... 109
Parameters ... 109
Example ... 109
Return Values ... 109
Storing NV Pairs in a Container ... 110
Syntax ... 110
Parameters ... 110
Example ... 110
Return Value ... 110
Searching for an NV Pair ... 110
Syntax ... 110
Parameters ... 110
Example ... 110
Return Values ... 110
Removing one NV Pair ... 111
Syntax ... 111
Parameters ... 111
Example ... 111
Return Value ... 111
Removing all NV Pairs from a Container ... 111
Syntax ... 111
Parameter ... 111
Example ... 111
Return Value ... 112

Totaling NV Pairs in a Container 112
Syntax 112
Parameter 112
Example 112
Return Value 112
Finding an NV Name 112
Syntax 112
Parameters 112
Example 112
Return Value 112
Finding an NV Value 113
Syntax 113
Parameters 113
Example 113
Return Value 113
Syntax 113
Parameters 113
Example 113
Return Value 114
Executing Business Logic Methods 114
Syntax 114
Parameters 114
Example 114
Return Values 114
Working with Multi-Value Name/Value Pairs 114
Determining the Type of a Name/Value Pair 114
Syntax 115
Parameters 115
Return Values 115
Determining the Position of an NV Container in an NV Hierarchy 115
Syntax 115
Parameters 115
Example 115
Syntax 116
Parameters 116
Example 116
Syntax 116
Parameters 116
Example 116
Getting the Number of Values in a Multi-Value Pair 116
Syntax 116
Parameters 116
Example 117
Retrieving Containers from an NV Pair 117
Syntax 117
Parameters 117
Example 117
Syntax 117
Parameters 117

Example ... 117
Retrieving the Values in a Multi-Value Pair ... 118
Syntax ... 118
Parameters ... 118
Example ... 118
Deleting Values from a Multi-Value Pair ... 118
Assigning a Container to a Parent ... 118
Syntax ... 119
Parameters ... 119
Example ... 119
Syntax ... 119
Parameters ... 119
Example ... 119
Syntax ... 120
Parameters ... 120
Example ... 120
Syntax ... 120
Parameters ... 120
Example ... 120
Appending String Values to a Multi-Value Pair ... 120
Syntax ... 120
Parameters ... 121
Example ... 121
Low-level Data Access & Manipulation ... 121
Reading Security and Rights for a DLL User ... 121
Syntax ... 121
Parameters ... 121
iOption values ... 121
Return Values ... 122
Syntax ... 122
Parameters ... 122
Return Values ... 122
Syntax ... 123
Parameters ... 123
Return Values ... 123
Returning GoldMine Licensing Information ... 123
Syntax ... 123
Parameters ... 123
Return Values ... 123
Notes ... 123
GMW_GetLicenseInfo Structure ... 123
Example ... 124
Returning Calendar Data ... 124
Syntax ... 124
Retrieving Data with DataStream ... 124
Advantages of Using DataStream ... 125
DataStream Record Selection ... 125
GMW_DS_Range ... 126
Syntax ... 126

Parameters 126
Return Values 126
GMW_DS_Range Field Selection 126
GMW_DS_Query 127
Syntax 127
Parameters 127
Return Values 127
GMW_DS_Fetch 128
Syntax 128
GMW_DS_Fetch Return Packet 128
GMW_DS_Close 129
Syntax 129
Accessing Low-Level Data Using Work Areas 129
GMXS32.DLL Low-Level Access Functions 129
Opening a Data File 131
Syntax 131
Parameter 131
Return Values 131
GMW_DB_Open Return Values 131
Closing a Data File 131
Syntax 131
Parameters 132
Return Values 132
GMW_DB_Close Return Values 132
Checking for an SQL Table 132
Syntax 132
Parameter 132
Return Values 132
GMW_DB_IsSQL Return Values 132
Adding a Record 132
Syntax 133
Parameters 133
Return Value 133
Deleting the Current Record 133
Syntax 133
Parameter 133
Return Values 133
GMW_DB_Delete Return Values 134
Querying for a Field Value 134
Syntax 134
Parameters 134
Checking the Current Record Number or Record ID 134
Syntax 134
Parameters 134
Return Value 135
Changing a Field Value 135
Syntax 135
Parameters 135
Return Values 135

Unlocking a Record 135
Syntax 135
Parameter 136
Return Values 136
GMW_DB_Unlock Return Values 136
Creating a Subset of Records 136
Syntax 136
Parameters 136
Return Values 136
Limiting Search Scope 136
Syntax 137
Parameters 137
Return Values 137
GMW_DB_Range Return Values 137
Performing a Sequential Search 137
Syntax 137
Parameters 137
Return Values 137
Moving to the First Record Match 138
Syntax 138
Parameters 138
Return Values 138
GMW_DB_Seek Return Values 138
Setting the Current Index Tag 138
Syntax 138
Parameters 139
Return Values 139
GMW_DB_SetOrder Return Values 139
Positioning the Record Pointer 139
Syntax 139
Parameters 139
GMW_DB_Move Commands and Function Equivalents 139
Return Values 140
GMW_DB_Move Return Values 140
Moving to a Specified Record 140
Syntax 140
Parameters 140
Return Values 140
GMW_DB_Goto Return Values 140
Moving to the First Record 141
Syntax 141
Parameter 141
Return Values 141
GMW_DB_TopReturn Values 141
Moving to the Previous or Following Record 141
Syntax 141
Parameters 141
Return Values 142
GMW_DB_Skip Return Values 142

Moving to the Last Record 142
Syntax 142
Parameter 142
Return Values 142
GMW_DB_Bottom Return Values 142
Seeking a Record 142
Syntax 143
Parameters 143
Return Values 143
Reading a Field Value 143
Syntax 143
Parameters 143
Return Values 144
Replacing a Field Value 144
Syntax 144
Parameters 144
Return Values 144
Updating Sync Logs with GMXS32.DLL 145
Updating the Sync Log File 145
Syntax 145
Parameters 145
Return Values 145
GMW_UpdateSyncLog Return Values 146
Example 146
Importing a Prepared TLog Import File 146
Syntax 146
Parameters 146
Return Values 146
Notes 147
TLog Import Structure 147
Example 147
Getting a New Record ID 147
Syntax 147
Parameters 147
Return Value 147
Notes 148
Example 148
Converting the Sync Stamp 148
Syntax 148
Parameters 148
Return Values 148
GMW_SyncStamp Return Values 148
Notes 148
Example 148
Working with the XML API 149
Overview 149
Executing Your XML Document 149
Example 149
Creating Your XML Document 149

Loading the API (GoldMine 7.0 or higher) ... 150
Parameters ... 150
LoadAPI Return Values ... 151
Loading BDE (GoldMine 6.7) ... 151
Parameters ... 152
LoadBDE Return Values ... 152
Logging in Subsequent Users ... 153
Parameters ... 153
Login Return Values ... 154
Logging Out ... 154
Syntax ... 154
Parameters ... 154
Return ... 154
Unloading the API (GoldMine 7.0 or higher) ... 154
Unloading BDE (GoldMine 6.7) ... 154
Accessing Data with Business Logic Functions ... 155
Accessing Nested Nodes of Data ... 155
Business Logic Function Return Values ... 155
Input XML: ... 155
Returned XML: ... 156
Accessing Low-level Data Manipulation Functionality ... 156
Retrieving Data with DataStream ... 156
Advantages of Using DataStream ... 156
DataStream Record Selection ... 157
DS_Range ... 157
Syntax ... 157
Parameters ... 157
Return Values ... 158
GMW_DS_Range Return Values ... 158
DS_Range Field Selection ... 158
DS_Query ... 158
Syntax ... 158
Parameters ... 158
Return Values ... 159
DS_Fetch ... 159
Syntax ... 159
Parameters ... 159
Optional Parameters ... 159
The XML Return packet ... 159
Return ... 160
DS_Fetch Return Packet ... 161
DS_Close ... 162
Syntax ... 162
Accessing Low-Level Data Using Work Areas ... 162
GMXS32.DLL Low-Level Access Functions ... 162
GMXS32.DLL Low-Level Access Functions ... 163
Opening a Data File ... 164
Syntax ... 164
Parameter ... 164

Return Values 164
DB_Open Code Attribute Values 164
Closing a Data File 164
Syntax 164
Parameters 164
Return Values 165
Checking for an SQL Table 165
Syntax 165
Parameter 165
Return Value 165
DB_IsSQL Code Attribute Values 165
Adding a Record 165
Syntax 165
Parameters 166
Return Value 166
Deleting the Current Record 166
Syntax 166
Parameter 166
Return Value 166
DB_Delete Code Attribute Values 166
Reading a Field Value 167
Syntax 167
Parameters 167
Return Value 167
DB_Range Code Attribute Values 167
Checking the Current Record Number or Record ID 167
Syntax 167
Parameters 167
Return Value 168
Changing a Field Value 168
Syntax 168
Parameters 168
Return Value 168
Unlocking a Record 169
Syntax 169
Parameter 169
Return Value 169
Creating a Subset of Records 169
Syntax 169
Note 169
Parameters 169
Return Value 170
DB_Filter Code Attribute Values 170
Limiting Search Scope 170
Syntax 170
Parameters 170
Return Value 170
DB_Range Code Attribute Values 171
Performing a Sequential Search 171

Syntax 171
Parameters 171
Return Value 171
DB_Search Code Attribute Values 171
Moving to the First Record Match 171
Syntax 172
Parameters 172
Return Value 172
DB_Seek Return Values 172
Setting the Current Index Tag 172
Syntax 172
Parameters 172
Return Value 173
DB_SetOrder Code Attribute Values 173
Positioning the Record Pointer 173
Syntax 173
Parameters 173
DB_Move Commands and Function Equivalents 173
Return Value 174
DB_Move Code Attribute Values 174
Moving to a Specified Record 174
Syntax 174
Parameters 174
Return Value 174
DB_Goto Code Attribute Values 175
Moving to the First Record 175
Syntax 175
Parameter 175
Return Value 175
DB_Top Code Attribute Values 175
Moving to the Previous or Following Record 175
Syntax 176
Parameters 176
Return Value 176
DB_Skip Code Attribute Values 176
Moving to the Last Record 176
Syntax 176
Parameter 176
Return Value 176
DB_Bottom Code Attribute Values 177
Seeking a Record 177
Syntax 177
Parameters 177
Return Value 177
DB_QuickSeek Code Attribute Values 177
Reading a Field Value 178
Syntax 178
Parameters 178
Return Value 178

DB_QuickRead Code Attribute Values ... 178
Replacing a Field Value ... 178
Syntax ... 179
Parameters ... 179
Return Value ... 179
DB_QuickReplace Code Attribute Values ... 179
Returning Calendar Data ... 179
Syntax ... 180
Return Value ... 180
Updating Sync Logs ... 181
Updating the Sync Log File ... 181
Syntax ... 181
Parameters ... 181
Return Value ... 181
UpdateSyncLog Code Attribute Values ... 182
Importing a Prepared TLog Import File ... 182
Syntax ... 182
Parameters ... 182
Return Value ... 182
ReadImpTLog Code Attribute Values ... 182
Notes ... 183
TLog Import Structure ... 183
Getting a New Record ID ... 183
Syntax ... 183
Parameters ... 183
Return Value ... 183
Notes ... 183
Converting the Sync Stamp ... 183
Syntax ... 183
Parameters ... 184
Return Value ... 184
SyncStamp Code Attribute Values ... 184
Notes ... 184
Using MSXML to Handle GoldMine API XML ... 184
Getting Started ... 184
Defining the Root Element ... 184
Setting Attributes ... 185
Referencing an Attribute ... 185
Creating Child Elements ... 185
Executing the XML Document ... 186
Reading the Results ... 187
Reading the Code Attribute ... 187
Reading the Returned Data ... 187
Accessing the Current GoldMine Instance with COM ... 189
Overview ... 189
Getting Started ... 189
Executing Commands ... 190
Logging In to GoldMine ... 190
GoldMine.UI Class ... 191

Accessing Data Files 191
Adding an Empty Record 191
Parameters 191
Return Value 191
Returned XML 191
Closing an Opened File 192
Parameters 192
Return Value 192
Returned XML 192
Deleting the Current Record 192
Parameters 192
Returned XML 192
Creating a Subset of Records 192
Parameters 193
Checking for an Xbase or SQL Table 193
Parameters 193
Return Value 193
Returned XML 193
Moving to a Specified Record 193
Parameters 194
Return Value 195
Move Return Values 195
Returned XML 195
Opening a Data File 195
Parameters 196
Open Valid Parameters 196
Return Value 196
Returned XML 196
Limiting GoldMine Search Range 196
Parameters 197
Returned XML 197
Parameters 197
Returned XML 197
Reading a Field Value 197
Parameters 197
Return Value 198
Returned XML 198
Checking the Current Record Number or Record ID 198
Parameters 198
Return Value 198
Returned XML 198
Changing a Field Value 199
Parameters 199
Return Value 199
Performing a Sequential Search 200
Parameters 200
Return Value 200
Returned XML 200
Parameters 201

Return Value 201
Returned XML 201
Accessing Contact Records 201
Differences in Accessing Contact Information 202
Parameters 202
Valid RecordObj Functions 202
Return Value 206
Returned XML 206
Accessing Specialized GoldMine.UI Functions 206
Retrieving a List of Active Plug-Ins (GoldMine 7.0 or higher) 206
Returned XML 206
Running a Plug-In (GoldMine 7.0 or higher) 207
Returned XML 207
Retrieving Login Credentials for Use with the GMXS32.DLL 207
Returned XML 207
Retrieving the RecID of the Current Opportunity 208
Return Value 208
Returned XML 208
Completing a Calendar Activity 208
Parameters 208
Return Value 209
Returned XML 209
Displaying Edit Windows for Calendar and History Items 209
General Messages 209
Return Value 209
Displaying the Contact Record of an Incoming Caller 210
Parameters 210
Return Value 211
CallerID Return Values 211
Returned XML 211
Running a Counter 211
Parameters 211
Return Value 211
Example 212
Returning GoldMine Record Data 212
Record Selection 213
Datastream Range Parameters 213
Datastream Query Parameters 213
Datastream Fetch Parameters 214
Datastream Close Parameters 214
The XML Return Packet 214
Returns 214
Return Packet 215
Performance 216
Processing a Web Import Instruction File 216
Reading an Xbase Expression Without Opening a File 217
Parameters 217
Return Value 217
Returns: 217

Adding Merge Fields to a Form 217
Parameters 217
Deleting Fields from a Form 218
Parameters 218
Return Value 218
Closing a Form Profile 218
Parameters 218
Creating an Xbase File with Registered Fields 218
Parameters 218
WhichRec Values 219
Return Value 219
Returning a Field Name for an Expression 219
Parameters 219
Returning a Value for Unattached Fields 219
Return Value 220
Counting the Number of Exported Records 220
Parameters 220
FormQueryCreate Parameters 220
Return Value 220
FormPrintedDoc 220
Parameters 220
Creating a History Record 221
Parameters 221
Return Value 222
Returned XML 222
Creating or Updating a Document Link 222
Parameters 222
Sync Valid Values 223
Return Value 223
Returned XML 223
Displaying a Message Dialog Box 223
Parameters 223
MsgBox Style Values 223
Return Value 224
Returned XML 224
Adding a Merge Form 225
Parameters 225
Document Types 225
Flag Values 226
Return Value 226
Playing a Toolbar Macro 226
Parameters 226
Identifying a Macro by Number 226
Identifying a Macro by File Name 226
Return Value 227
PlayMacro Return Values 227
Optional switches include: 227
Creating and Sending a Pager Message 227
Return Value 228

Displaying a Message in the GoldMine Status Bar 228
Parameters 228
Returned XML 228
Converting TLog Timestamps 228
Parameter 229
Return Value 229
Returned XML 229
Updating the Sync Log File 229
Parameters 229
Return Value 229
UpdateSyncLog Code Attribute Values 229
Importing a Prepared TLog Import File 230
Syntax 230
Parameters 230
Return Value 230
ReadImpTLog Code Attribute Values 230
Notes 230
TLog Import Structure 230
Forcing Logout 231
Syntax 231
Parameters 231
Reading Security and Rights 231
Syntax 231
Permissions Returned by UserAccess 231
Returned XML 232
Retrieving Calendar Permissions 233
Syntax 233
Parameters 233
Return Value 233
Retrieving History Access 233
Syntax 233
Parameters 233
Return Value 233
Macros 234
Executing Macros 234
Returned XML 234
Available Data-Related Macros 234
Macros for Merge Forms 242
&PARAM2 Parameters 242
&PARAM3 Parameters 243
&PARAM4 Parameters 243
&PARAM5 Parameters 244
Macros for the GoldMine License 244
Controlling the GoldMine User Interface 245
Getting Window Information 245
GetAvailableWindowsList 245
Syntax 245
Returned XML 246
GetActiveWindowsList 246

Syntax 246
Returned XML 246
Registering for Events 247
RegisterVetoWindowLaunch 248
Syntax 248
Parameters 248
Returned XML 248
RegisterWindowUpDown 248
Syntax 248
Parameters 248
REturned XML 249
RegisterCommandExec 249
Syntax 249
Parameters 249
Returned XML 249
RegisterTabDetailsEvent 250
Syntax 250
Parameters 250
AdditionalContactClick 251
AdditionalContactClick 251
Returned XML 251
Parameters 251
DetailsClick 251
Returned XML 251
Parameters 252
PendingClick 252
Returned XML 252
Parameters 252
HistoryClick 252
Returned XML 252
Parameters 252
LinkedDocClick 253
Returned XML 253
Parameters 253
Handling GoldMine.UI Events 253
NotifyControlCommand 253
Parameters 253
VetoWindow 253
Parameters 254
Example 254
WindowUpDown 255
Parameters 255
GMEvent 255
Returns 256
Manipulating Controls Programatically 257
PressButton 257
Syntax 258
Parameters 258
SetControlText 258

Syntax ... 258
Parameters ... 259
SetCheckBox ... 259
Syntax ... 259
Parameters ... 259
SelectRadio ... 259
Syntax ... 260
Parameters ... 260
SetListBox/SetComboBox ... 260
Syntax ... 261
Parameters ... 261
SelectTab ... 261
Syntax ... 262
Parameters ... 262
EnableCtrl ... 262
Syntax ... 263
Parameters ... 263
Executing a Menu Command ... 263
Syntax ... 263
Returned XML ... 266
Opening a Mail Record ... 266
Syntax ... 266
Parameters ... 266
Returned XML ... 267
Setting a Selected Record in a GoldMine Grid (GoldMine 8.0 or higher) ... 267
Parameters ... 267
Returned XML ... 267
Returning Selected Records in a GoldMine Grid (8.0.1 or higher) ... 268
Syntax (Example) ... 268
Parameters ... 268
Returned XML ... 268
GoldMine.RecObj Class ... 268
RecordObjectHasChanged ... 269
Parameters ... 269
RecordFieldHasUpdated ... 269
Parameters ... 269
RecordTabHasChanged ... 269
Parameters ... 269
GoldMine.GMSystemEvents Class ... 269
GoldMineshutDown ... 269
Business Logic Methods ... 271
Overview ... 271
Business Logic Functions and Name/Value Pairs ... 271
Controlling Database Session Handling ... 271
Creating or Updating a Contact Record ... 271
Required Name/Value Pairs ... 272
Optional Name/Value Pairs ... 272
Special Name/Value Pairs ... 272
Output Name/Value Pairs ... 272

WriteCONTACT Error Codes 272
Updating an E-mail Address 273
Required Name/Value Pairs 273
Optional Name/Value Pairs 273
Updating a Web Site Record 274
Name/Value Pairs 274
Updating Notes of a Primary Contact Record 274
Required Name/Value Pairs 274
Optional Name/Value Pairs 274
Output Name/Value Pairs 274
Creating or Updating a Note in a Table 274
Required Name/Value Pairs 275
Optional Name/Value Pairs 275
Output Name/Value Pairs 275
WriteNote Error Codes 275
Creating or Updating an Additional Contact Record 276
Required Name/Value Pairs 276
Optional Name/Value Pairs 276
Special Name/Value Pairs 277
Error Codes 277
Output Name/Value Pairs 277
Creating or Updating a Detail Record 278
Required Name/Value Pairs 278
Optional Name/Value Pairs 278
Special Name/Value Pairs 278
Output Name/Value Pairs 278
Error Codes 278
Creating or Updating a Linked Document 279
Required Name/Value Pairs 279
Optional Name/Value Pairs 279
Special Name/Value Pairs 279
Output Name/Value Pairs 279
Error Codes 279
Creating or Updating a Referral 280
Required Name/Value Pairs 280
Optional Name/Value Pairs 280
Special Name/Value Pairs 280
Output Name/Value Pairs 280
Creating or Updating Activities 281
Required Name/Value Pairs 281
GoldMine 6.0 NV Pairs 283
Optional WriteSchedule NV Pairs 283
Output Name/Value Pairs 285
Error Codes 285
Creating or Updating a History Record 285
Required Name/Value Pairs 285
WriteHistory Optional Name/Value Pairs 285
WRITE HISTORY Special Name/Value Pairs 286
Output Name/Value Pairs 286

Creating or Updating a Case Record (GoldMine 8.0 or higher) 286
Required Name/Value Pairs 286
Optional Name/Value Pairs 287
Error Codes 288
Output Name/Value Pairs 288
Creating or Updating a Case Attachment (GoldMine 8.0 or higher) 288
Required Name/Value Pairs 288
Optional Name/Value Pairs 288
Error Codes 289
Output Name/Value Pairs 289
Adding a GoldMine User as a Case Team Member (GoldMine 8.0 or higher) 289
Required Name/Value Pairs 289
Error Codes 290
Output Name/Value Pairs 290
Attaching an Automated Process 290
ATTACHTRACK Required Name/Value Pairs 290
Output Name/Value Pairs 291
Executing an SQL Query 291
Required Name/Value Pairs 291
Optional Name/Value Pairs 291
Output Name/Value Pairs 291
Creating a Cont act Group 292
Required Name/Value Pairs 292
Optional Name/Value Pairs 292
Output Name/Value Pairs 292
Return Codes 292
Adding Contacts to a Contact Group 293
Required Name/Value Pairs 293
Members NV Pair Child Container Name/Value Pairs 293
Output Name/Value Pairs (parent container) 293
Return Codes 293
Using AddContactGrpMembers 294
Reading a Record 294
Required Name/Value Pairs 294
Optional Name/Value Pairs 295
Special NVs 295
Output Name/Value Pairs 295
Return Codes 295
Reading a Contact1 or Contact2 Record 295
Required Name/Value Pairs 295
Optional Name/Value Pairs 296
Special NVs 296
Output Name/Value Pairs 296
Return Codes 296
Returning Alerts Attached to a Contact Record 296
Required Name/Value Pairs 296
Output Name/Value Pairs 297
Return Codes 297
Attaching an Alert 297

Required Name/Value Pairs 297
Output Name/Value Pairs 298
Returning All Alerts 298
Required Name/Value Pairs 298
Output Name/Value Pairs 298
Required Name/Value Pairs 299
Output Name/Value Pairs 299
Returning a User Group Member List 299
Required Name/Value Pairs 299
Output Name/Value Pairs 299
Returning Group Memberships for a Specified User 299
Required Name/Value Pair 299
Output Name/Value Pairs 300
Saving a User Group 300
Required Name/Value Pairs 300
Output Name/Value Pair 300
Retrieving the Names of User Groups 300
Required Name/Value Pairs 300
Return Name/Value Pairs 300
Example 301
Evaluating an Xbase Expression on a Contact Record 301
Name/Value Pairs 301
Return Values 302
Encrypting Text 302
Required Name/Value Pairs 302
Decrypting Encoded Text 303
Required Name/Value Pairs 303
Returned Name/Value Pairs 303
Retrieving the Default Contact Automated Process 303
Deleting Calendar Items 303
Deleting History Items 304
Required Name/Value Pairs 304
Return Values 304
Handling GoldMine Security 305
Creating a New GoldMine Login 305
Name/Value Pairs 305
Return Values 305
Reading a GoldMine Login 305
Output Name/Value Pairs 305
Return Values 306
Retrieving Security Access 306
Retrieving Field-Level Access Rights 308
Required Name/Value Pairs 308
Example NV Container Returned from FieldAccessRights 308
Retrieving Visible Fields 308
Checking for Record Curtaining 309
Required Name/Value Pairs 309
Output Name/Value Pair 309
Name/Value Pairs 309

Return Name/Value Pairs 310
Removing a Remote License 310
Name/Value Pairs 310
Return Name/Value Pairs 310
E-mail Name/Value Functions 311
Reading a Mail Message 311
Required Name/Value Pairs 311
Optional Name/Value Pairs 311
READMAIL Output Name/Value Pairs 311
Queuing a Message for Delivery 314
QueueMail Optional NV Pairs 314
Return Name/Value Pairs 315
Required Name/Value Pairs 316
Optional Name/Value Pairs 316
Optional Name/Value Pairs 316
Return Codes 317
Deleting a Message 317
Required Name/Value Pairs 317
Filing a Message in History 317
Required Name/Value Pairs 318
Optional Name/Value Pairs 318
Return Codes 318
Preparing the NV Container for a New Mail Message 318
Required Name/Value Pairs 318
Optional Name/Value Pairs 319
Return Name/Value Pairs 319
Preparing the NV Container to Reply to a Mail Message 319
Required Name/Value Pairs 319
Optional Name/Value Pairs 319
Return Name/Value Pairs 320
Preparing an NV Container to Forward a Mail Message 320
Required Name/Value Pairs 320
Optional Name/Value Pairs 320
Return Name/Value Pairs 320
Adding an E-mail Center Folder 321
Name/Value Pairs 321
Deleting an E-Mail Center Folder 321
Name/Value Pairs 321
Obtaining a List of E-Mail Center Folders 321
Return Name/Value Pairs 321
Return Name/Value Pairs 322
Accessing E-mail Templates 322
Optional Name/Value Pairs 322
Return Name/Value Pairs 322
Retrieving E-mail Account Information 323
Return Name/Value Pairs 323
Retrieving a List of Messages Waiting Online 324
Required Name/Value Pairs 324
Return Name/Value Pairs 324

Return Values 326
Retrieving Messages 326
Required Name/Value Pairs 326
Return Name/Value Pairs 326
Return Values 327
Deleting Online E-mail Messages 327
Required Name/Value Pairs 327
Return Name/Value Pairs 327
Return Values 327
Saving a Manual List of Recipients 328
Retrieving a Manual List of Recipients 328
Managing Internet E-mail Preferences 328
Optional input (SetEmailPrefs) and Output (GetEmailPrefs) Name/Value Pairs 328
Profiles child containers have the following NV Pairs. 332
Required Name/Value Pairs 333
Special Name/Value Pairs 334
Output Name/Value Pairs 334
Notes 334
Manipulating User-Defined Fields and Views 334
Reading All Field Views 335
Output Name/Value Pairs 335
VIEW Name/Value Pairs 335
Field Name/Value Pairs 335
GetContactViews Return Values 336
Deleting a Contact View 337
DeleteContactViews Return Values 337
Creating or Modifying a Contact View 337
input Name/Value Pairs 337
Field Name/Value Pairs 338
WriteContactView output NV pairs 339
WriteContactView Return Values 339
Reading Custom Fields 340
ReadCustomFields input NV pairs 340
Field NV Pair Container 340
ReadCustomfields Return Values 340
Modifying the Structure of Custom Fields 341
EditCustomField Input NV pairs 341
EditCustomField Return Values 341
Reading Calendar Preferences 342
READCALENDARPREFS Input NV pairs 342
READCALENDARPREFS OUTPUT NV pairs 342
READCALENDARPREFS RETURN VALUES 348
Modifying Calendar Preferences 349
WRITECALENDARPREFS Input NV pairs 349
WRITECALENDARPREFS OUTPUT NV pairs 349
WRITECALENDARPREFS RETURN VALUES 355
Reading Personal Preferences 355
READPERSONALPREFS Input NV pairs 356
READPERSONALPREFS OUTPUT NV pairs 356

READPERSONALPREFS RETURN CODES ... 356
Updating Personal Preferences ... 356
WRITEPERSONALPREFS Input NV pairs ... 356
WRITEPERSONALPREFS OUTPUT NV pairs ... 356
WRITEPERSONALPREFS RETURN CODES ... 357
Reading Record Preferences ... 357
READRECORDPREFS Input NV pairs ... 357
READRECORDPREFS OUTPUT NV pairs ... 357
READRECORDPREFS RETURN CODES ... 358
Updating Record Preferences ... 358
WRITERECORDPREFS Input NV pairs ... 358
WRITERECORDPREFS RETURN CODES ... 359
Reading Schedule Preferences ... 359
READSCHEDULEPREFS Input NV pairs ... 359
READSCHEDULEPREFS OUTPUT NV pairs ... 360
READSCHEDULEPREFS RETURN CODES ... 360
Updating Schedule Preferences ... 360
WRITESCHEDULEPREFS Input NV pairs ... 360
WRITESCHEDULEPREFS RETURN CODES ... 361
Reading Alarm Preferences ... 361
READALARMPREFS Input NV pairs ... 361
READALARMPREFS OUTPUT NV pairs ... 361
READALARMPREFS RETURN CODES ... 362
Updating Alarm Preferences ... 362
WRITEALARMPREFS Input NV pairs ... 362
WRITEALARMPREFS RETURN CODES ... 362
Reading Lookup Preferences ... 363
READLOOKUPPREFS Input NV pairs ... 363
READLOOKUPPREFS OUTPUT NV pairs ... 363
READLOOKUPPREFS RETURN CODES ... 363
Updating Alarm Preferences ... 364
WRITELOOKUPPREFS Input NV pairs ... 364
WRITELOOKUPPREFS Return Codes ... 364
Reading Pager Preferences ... 364
READPAGERPREFS Input NV pairs ... 364
READPAGERPREFS OUTPUT NV pairs ... 365
READPAGERPREFS Return Codes ... 365
Updating Pager Preferences ... 365
WRITEPAGERPREFS Input NV pairs ... 365
WRITEPAGERPREFS Return Codes ... 366
Reading Miscellaneous Preferences ... 366
READMISCPREFS Input NV pairs ... 366
READMISCPREFS OUTPUT NV pairs ... 366
READMISCPREFS Return Codes ... 367
Updating Miscellaneous Preferences ... 367
WRITEMISCPREFS Input NV pairs ... 367
WRITEMISCPREFS Return Codes ... 367
Reading the Database Engine Type (7.0 or higher) ... 368
GETDBENGINETYPE Return Codes ... 368

Reading a List of GoldMine User Groups ... 368
GETGMUSERGROUPS OUTput NV pairs ... 368
GETGMUSERGROUPS Return Codes ... 368
Creating or Updating GoldMine User Groups ... 369
WRITEGMUSERGROUP Input NV pairs ... 369
WRITEGMUSERGROUP Return Codes ... 369
Adding a GoldMine User to a Group ... 369
ADDGMGROUPUSER Input NV pairs ... 369
ADDGMGROUPUSER Return Codes ... 370
Removing a GoldMine User from a Group ... 370
REMOVEGMGROUPUSER Input NV pairs ... 370
REMOVEGMGROUPUSER Return Codes ... 370
Creating or Updating an Opportunity or Project ... 371
WRITEOPPROJ Input NV pairs ... 371
WRITEOPPROJ Return Codes ... 372
Working with GoldMine Plug-ins ... 373
Overview ... 373
Using ActiveX Plug-in Support ... 373
Using HTML Plug-in Support ... 374
Plug-In Description File ... 374
HTML Plug-in Description File ... 374
ActiveX Plug-in Description File ... 376
Security and Plug-in Directories ... 378
Security ... 378
Adding a Local Plug-in Directory ... 379
Sample Plug-ins ... 379
gmail.gme ... 379
External.gme ... 380
gmplus.asp ... 380
Using Xbase Expressions ... 384
Overview ... 384
Function/Parameter Types ... 384
Conditionals, Operators, and Logical Evaluators ... 385
Conditionals ... 385
Operators ... 387
Logical Evaluators ... 388
Xbase Functions ... 388
String Functions ... 389
Date Functions ... 392
Numeric Functions ... 394
Miscellaneous Functions ... 396
Xbase Database Structures ... 398
Overview ... 398
CAL.DBF ... 399
CAL Indexes ... 399
CAL Structure ... 399
Rectype ... 400
CONTACT1.DBF ... 401
CONTACT1 Indexes ... 401

CONTACT1 Relations 401
CONTACT1 Structure 402
Account Number 403
Internal Status 403
CONTACT2.DBF 404
CONTACT2 Index 404
CONTACT2 Structure 404
CONTGRPS.DBF 405
CONTGRPS Indexes 405
CONTGRPS Structure (header records) 405
Header Info 406
CONTGRPS Structure (member records) 406
CONTHIST.DBF 406
CONTHIST Indexes 406
CONTHIST Structure 407
Record Type 408
CONTSUPP.DBF 408
CONTSUPP Indexes 408
CONTSUP Structure 408
Record Type 409
INFOMINE.DBF 410
INFOMINE Indexes 410
INFOMINE Structure 410
LOOKUP.DBF 411
LOOKUP Indexes 411
LOOKUP Structure 411
MAILBOX.DBF 411
MAILBOX Indexes 412
MAILBOX Structure 412
Flags 412
Folder 413
OPMGR.DBF 413
OPMGR Structure 413
Record Type 414
PERPHONE.DBF 414
PERPHONE Indexes 415
PERPHONE Structure 415
RESITEMS.DBF 415
RESITEMS Indexes 415
RESITEMS Structure 415
SPFILES.DBF 416
SPFILES Index 416
SPFILES Structure 416
SQL Database Structures 417
Overview 417
CAL Table 417
CAL Indexes 418
CAL Structure 418
Record Type 419

CONTACT1 Table 419
CONTACT1 Indexes 419
CONTACT1 Relations 420
CONTACT1 Structure 420
Account Number 422
Internal Status 423
CONTACT2 Table 423
CONTACT2 Index 423
CONTACT2 Structure 423
CONTGRPS Table 424
CONTGRPS Indexes 424
CONTGRPS Structure (header records) 424
Header Info 425
CONTGRPS Structure (member records) 425
CONTHIST Table 425
CONTHIST Indexes 425
CONTHIST Structure 426
Record Type 427
CONTSUPP Table 427
CONTSUPP Indexes 427
CONTSUPP Structure 427
Record Type 428
INFOMINE Table 429
INFOMINE Indexes 429
INFOMINE Structure 429
LOOKUP Table 430
LOOKUP Indexes 430
LOOKUP Structure 430
MAILBOX Table 430
MAILBOX Indexes 430
MAILBOX Structure 431
Flags 431
Folder 431
OPMGR Table 432
OPMGR Indexes 432
OPMGR Structure 432
Record Type 433
PERPHONE Table 433
PERPHONE Indexes 433
PERPHONE Structure 434
RESITEMS Table 434
RESITEMS Indexes 434
RESITEMS Structure 434
SPFILES Table 434
SPFILES Index 435
SPFILES Structure 435
Appendix: Code Examples 436
Overview 436
GMXS32.DLL Code Examples 436

C++ Examples 436
Function prototypes 436
Logging In 438
Creating a Contact with Business Logic/Enumerating a Name Value Container/DataStream 439
Low-Level Work Area 441
Visual Basic Examples 442
Function prototypes 442
Logging In 445
Creating a Contact 445
Enumerating a Container 446
DataStream 446
Low-Level WorkArea 447
Delphi Examples 448
Function prototypes 449
Creating a Contact 452
Enumerating a Container 452
DataStream 453
Low-Level Work Area 453
Resources 455
Additional Documentation 455
Contact Us 455
Support Site 455
Contact Information 455
Index 456

Introduction to Integrating with GoldMine

Introduction

Integrating with GoldMine is designed as a comprehensive resource for developers to integrate GoldMine with their applications. For best results, we recommend that you become an experienced GoldMine user before taking on an integration project. For example, understanding what types of data are better stored as a detail record instead of a history record will ensure greater success for your project.

In addition to gaining experience with GoldMine, you should be familiar with the development environment you plan to use. This manual may not provide programming examples for your preferred development environment. With a good working knowledge of your chosen programming language, you could learn from another language's examples.

This manual provides information to:

- Use one of several methods to integrate with GoldMine.
- Work with either Xbase or SQL database structures to integrate with GoldMine up to version 6.7.
- Work with either Firebird or MSSQL database structures to integrate with GoldMine version 7.0.
- Access a variety of support resources to get help from other developers and GoldMine technicians.

IMPORTANT:

As of GoldMine 2018.2, and the introduction of forced password complexity, any integration that passes the GoldMine password via the GoldMine API must provide it "as-is." For previous versions (2018.1 and lower), the password had to be provided in UPPERCASE. This is *not* the case with GoldMine 2018.2 onward.

For example, with the user password of Access!123: In versions of GoldMine 2018.1 and lower, it was necessary to enter it as ACCESS!123. With GoldMine 2018.2 onward, it is necessary to pass the password as-is (i.e., Access!123).

Methods of Integrating with GoldMine

There are several methods for integrating with GoldMine:

- Dynamic Data Exchange (DDE)
- `GMXS32.DLL`
- `GMXMLAPI.DLL`
- GoldMine COM Server

- GoldMine Plug-ins (GoldMine 7.0 or higher)
- Database engine

Integrating via Dynamic Data Exchange

This method is supported by many programming environments, such as C++, Delphi, Visual Basic, VBA (Office 97—Access, Word, and Excel), WordBasic, FoxPro, and many others. DDE commands can be sent to GoldMine to make GoldMine perform a large variety of functions.

Integrating via GMXS32.DLL

You can also integrate with GoldMine using the GMXS32.DLL (The X represents the main version of GoldMine being used (i.e., 6 for GoldMine 6.0). Using the DLL method, you can access or maintain your GoldMine data without running GoldMine.

This DLL has enough functions for data access and synchronization maintenance to allow nearly full control of all databases and their fields. High-level "business logic" functions streamline and simplify performing common tasks, such as adding a contact, scheduling an activity, and so forth. GMXS32.DLL is placed into your Windows\System directory, and is updated automatically when you update GoldMine. This DLL does not require a separate license to use.

NOTE: This method of integration is highly recommended as it automates the task of adhering to GoldMine business logic rules, security, and synchronization.

Integrating via the GoldMine XML API (GMXMLAPI.DLL)

Another integration method, introduced in GoldMine 6.7, is the GoldMine XML API. This DLL allows the programmer to pass the GoldMine API an XML document to integrate with GoldMine. This API is another access method to the high-level business logic methods and the lower level data functions. The XML API is a COM object that can easily be used in various programming languages, including in the development of web applications. Using the versatile XML standard, integrating with GoldMine has never been easier.

Interacting with GoldMine via the GoldMine COM Server

With the release of GoldMine 6.7, a new method of interacting with a running GoldMine was introduced, the user-interface API. GoldMine is now a COM server. This method of interaction with GoldMine replaces the DDE functionality. DDE is still present in GoldMine for legacy integrations, but the new improved COM server capability adds a wealth of functionality that enables the programmer to control the GoldMine user-interface like never before. In addition, accessing GoldMine as a COM server is much easier than DDE in a .Net programming environment.

Integrating via GoldMine Plug-ins

GoldMine 7.0 contains a new mechanism to support ActiveX controls and HTML based integrations as if they were a part of GoldMine. These structures allow for rapid integration, ease of use, and security.

Integrating via a Database Engine

The most difficult method of integration involves writing to GoldMine databases via a database engine. Using this method also involves some work with DLL or DDE to keep GoldMine synchronization information intact. We do not recommend using this method because there is a higher likelihood of incorrect implementation, which could damage GoldMine data.

TIP: For best results, do not integrate via a database engine.

Comparing Integration Methods

The following table summarizes the integration methods and whether they require loading the Borland Database Engine, if GoldMine needs to be running, and if they require a GoldMine seat. Use this table to help determine the integration methods that best suits your application needs.

API Method	Requires BDE to be loaded?	Requires GoldMine to be running?	Uses seat?	Best used for
GMXS32.DLL	Yes	No	No	Perhaps highest speed, broad range of functionality
DDE	No	Yes	No	Minimal coding, slow speed, less functionality, only way in older GoldMine's of interfacing with GoldMine user interface
GoldMine COM Server (GoldMine.UI, GoldMine.RecObj, & GoldMine.SysEvents	No	Yes	No	Used for interacting with GoldMine user interface and also provides lower level functions. DDE replacement with much enhanced user interface control. Requires GoldMine to be running.
GoldMine COM Server (GoldMine.GoldMineData)	No	Yes	No	Broader range of functionality with business logic and lower level functions. Does not require BDE to be loaded. Alleviates SharedMemLocation errors commonly found with the GMXS32.DLL.
GMXMLAPI.DLL	Yes	No	Yes	Provides same functionality as the GMXS32.DLL, but provides easier XML interface

API Method	Requires BDE to be loaded?	Requires GoldMine to be running?	Uses seat?	Best used for
GoldMine Plug-ins	No	Yes	No	Provides a platform for developing GoldMine applications. Supports integrations developed using ActiveX Controls or HTML. Very powerful when used in conjunction with GoldMine APIs.
Direct Access through data engine (ex. ADO)	No	No	No	NOT RECOMMENDED!! Does not respect GoldMine security, does not automatically log synchronization information, does not have functionality to generate AccountNo's or Recid's, does not return encrypted GoldMine data in a readable format, requires intimate knowledge of GoldMine data rules.

NOTE: As of GoldMine Version 7.0, the Borland Database Engine is no longer used. References to BDE in the following table apply to integrations developed in GoldMine Version 6.7 or lower.

NOTE: As of GoldMine 2018.2, and the introduction of forced password complexity, GoldMine passwords must now be passed in proper case or "as-is" in the code when integrating with GoldMine. Any references that the user's password must be in UPPERCASE or case insensitive only apply to integrations developed in GoldMine versions 2018.1 or lower.

Resources and Support

In addition to this manual, GoldMine provides a variety of free resources to support developers, including:

- API/Programming topics on the GoldMine Forum
- Open Developer Community

Technology Partner Program

For specific questions and additional information, go to the GoldMine Community Forum at:

https://www.goldmine.com/community/

Experienced developers can offer advice or programming help. The newsgroup also contains advanced or hard-to-find information. This newsgroup is a self-serve resource and is not monitored or contributed to by GoldMine.

Open Developer Community

This online self-service resource provides technical documents, code samples, development tools, the most up-to-date documentation, and a searchable knowledgebase containing integration information.

Technology Partner Program

The Certified Technology Partner Program is intended for developers who wish to create and market products that integrate with our GoldMine and Ivanti products. These partners seek a close development, marketing, and sales relationship with GoldMine Inc.

Members of the Certified Technology Partner Program pay an annual fee and receive additional benefits over the Open Developer Community, including:

- Certification of your integrated solution (additional fees may apply for multiple certifications)
- Use of GoldMine and Ivanti Technology Partner logos to promote your product
- Listing on the Ivanti.com website
- Right to participate in beta programs
- Not-for-resale (NFR) licenses of GoldMine and Ivanti products
- Discounted product training
- Free and fee-based marketing programs

Integration Tools

The following tools can help when integrating with GoldMine:

- *DDERequestor*: A Windows-based freeware that allows you to send DDE commands to GoldMine in real-time. This utility can help to diagnose problems you may have when using DDE to integrate with GoldMine.
- *XMLSPY*: A development environment for modeling, editing, debugging, and transforming all XML technologies, then automatically generating runtime code in multiple programming languages.

NOTE: Technical support for these programs is not available from GoldMine.

Working with Dynamic Data Exchange (DDE)

Overview

Dynamic Data Exchange (DDE) is the term for the Windows functionality that allows GoldMine to exchange commands and information with other applications. Using DDE, one application, referred to as the client application, can request information from or send commands to another application—referred to as the server application. The server application then processes the request from the client application. In response to a client's request, the server performs a task such as updating or returning data housed by the server application

GoldMine is designed to act as both a DDE client as well as a DDE server. DDE topics included in this chapter describe using GoldMine as a DDE server. These topics are provided for programmers who wish to interface their programs with GoldMine. If you are not familiar with working with DDE, this technical section may be of limited value to you.

Using DDE in GoldMine

GoldMine can perform a variety of tasks using DDE commands, including:

- Merging data into a document
- Updating database information
- Querying for data
- Identifying telephone numbers automatically
- Linking contact records to an accounting application
- Inserting incoming e-mail

Merging Data into a Document

GoldMine uses DDE to communicate with your word processor. When you perform a merge, GoldMine uses DDE to send contact information to the word processor of the selected document template. The word processor receives this information from GoldMine, places the information from the contact record in appropriate places in the document, and then prints the document.

GoldMine acts as a DDE client and a DDE server during the document merging process. First, GoldMine must send a DDE request to the word processor to request that the word processor open a particular document template. Once the document is open, the word processor will recognize that the document contains DDE linkage fields and will ask GoldMine for data to place in these fields. GoldMine, now acting as a DDE server, will return this information to the word processor, and the word processor will update its display with the information. Finally, the document can be printed.

This type of merging can also be performed with other Windows applications, such as spreadsheets (for example, Microsoft Excel) or database programs (for example, Microsoft Access).

Updating Database Information

DDE can also be used to update GoldMine databases from another application. For example, a magnetic card reader application that supports DDE can be interfaced with GoldMine in such a way that new contact records are automatically entered into the contact database. Therefore, whenever a trade show attendee's badge is swiped through the reader, GoldMine is automatically updated.

Querying for Data

The DDE macros and other functions can query the GoldMine tables and return the contents to the caller. The [DataStream] command is a high-performance feature that can return large blocks of data very quickly. Retrieving data from large databases may take longer, causing your DDE request to time-out.

Identifying Telephone Numbers Automatically

GoldMine DDE functionality can be used with CallerID or ANI equipment to automatically identify incoming telephone calls. GoldMine can display the contact record that matches the telephone number of the incoming call, saving the user time in looking up the caller.

Linking Contact Records to an Accounting Application

DDE applications can be created to automatically transfer prospect information to an accounting application when the prospect decides to purchase, saving data entry time and reducing errors.

Inserting Incoming E-mail

DDE can be used to insert incoming e-mail into GoldMine, allowing GoldMine users to remain linked with their external e-mail systems.

Linking GoldMine to MS Word for Windows

The GoldMine DDE interface works with any Windows application that supports DDE; however, every application uses a unique format for executing DDE calls and for responding to DDE requests. Explaining all of the various methods to use DDE is beyond the scope of this manual. Instead, this document explores the use of DDE between GoldMine and another popular Windows application, Word 97 for Windows. The examples presented should provide a framework for creating DDE links to other applications.

NOTE: For details on installing the GoldMine DDE link to Word for Windows, see related material at: http://www.goldmine.com.

Entering Application, Topic, and Item Names

To establish a DDE conversation with an application that supports DDE, you must know the application's service name. The GoldMine service name is GoldMine.

GoldMine supports two service topics:

- *SYSTEM*: Queries a DDE server on supported data formats—for more information, see your Microsoft DDE documentation.
- *DATA*: Accesses all GoldMine DDE functions.

Specific GoldMine DDE functions are accessed by passing a DDE item string to GoldMine. The item can be a macro, a command, or an expression.

DDE Parameters, Functions, Expressions, Macros

Service	Topic	Item
GOLDMINE	SYSTEM	<item>
GOLDMINE	DATA	&<macro>
GOLDMINE	DATA	<expression>
GOLDMINE	DATA	[<function>]

GoldMine DDE functions can process a variety of tasks, including database query and manipulation. Commands are always passed surrounded by brackets. DDE functions are listed in Working with DDE Functions.

GoldMine can evaluate Xbase expressions by passing the expression as a DDE function call. For example, the expression *CONTACT1->CONTACT* will return the contact name of the current contact record displayed in the currently active contact record.

When a DDE item begins with an ampersand (&), GoldMine assumes that this item is a macro, and performs a lookup into an internal macro expansion table. If a match is found, GoldMine evaluates the macro and returns the result.

TIP: For a list of GoldMine DDE macros and their functions, see DDE Macros.

Establishing a DDE Conversation

The following example illustrates using Visual Basic for Applications (VBA) to establish a DDE conversation.

```
ch = DDEInitiate("GOLDMINE","DATA")
```

The DDEINITIATE function is used to establish the DDE link. The first parameter is the GoldMine service name; the second parameter is the service topic on which this DDE conversation is based. If the call is successful, the function returns a nonzero channel number to be used for all subsequent DDE requests to that channel. This channel number should not be confused with the work area pointer that GoldMine uses for many DDE functions.

If the DDEINITIATE function returns 0, the conversation could not be established.

Note that the examples within this chapter are written in Visual Basic for Applications, and the DDEInitiate and DDERequest functions are not a part of Visual Basic 6.0. DDE functionality is performed via the LinkRequest method in a textbox. The following example illustrates how the DDE conversation is initiated and requests are made in Visual Basic 6.0. The code can be written into a form that never gets displayed (only loaded) and be included in any of your VB projects.

To Initiate a DDE Conversation

```
Public Function DDEInitiate() As Integer

On Error GoTo Err_DDE

With txtGMDDE
.LinkMode = vbLinkNone
.LinkTopic = "GoldMine|Data"
.LinkMode = vbLinkManual
End With

DDEInitiate = 1

Exit Function

Err_DDE:
If Err = 282 Then
DDEInitiate = 282
Else
Err.Description = "DDE Error:" & Err & " :" & Err.Description
DDEInitiate = 0
End If

End Function
```

To Request Data

```
Public Function DDERequest(sExpr As String) As String

With txtGMDDE
.LinkItem = sExpr
.LinkRequest
DDERequest = .Text
End With

End Function
```

With these functions declared in your project, you may then call them where needed in your code.

Working with DDE Functions

GoldMine supports a variety of DDE functions, which are described in this section. Each function description includes calling format, description of operation, and an example of a VBA subroutine using the function.

GoldMine DDE functions allow access to other files or functions. Three categories of DDE functions provide access to the following:

- Data files
- Records
- Specialized functions

Depending on the type of application involved, you would typically select one of these three access methods; however, you can mix all three access methods within the same application. The function categories are described on the following pages.

Accessing Data Files

GoldMine provides a complete set of DDE functions that allow low-level access to the data files. These functions allow you to:

- Open particular data files,
- Query the values of the fields in the records in the data files,
- Add records to the files, and
- Replace data in the records.

This suite of functions is usually used for database applications that need varied access to GoldMine data.

Adding an Empty Record

Syntax	[APPEND(<work area>)]

The Append function is used to add an empty record to a GoldMine data file. Before using Append, you must open a data file using the Open function. After executing the Append function, the record pointer is positioned at the new empty record, and the record is locked and ready to accept field replacements.

When a CONTACT1 record is appended, GoldMine automatically propagates the new record with the appropriate ACCOUNTNO and CREATEBY values. For all other records, you must replace the ACCOUNTNO field with the value from the CONTACT1 record with which the new record is to be linked. For records that require remote synchronization initialization, GoldMine will automatically propagate the value of the RECID field when these records are appended.

Parameters

The Append function accepts one parameter, the work area handle of the file to Append. The work area handle is returned by the Open file when the file is opened.

Return Value

- *Xbase*: The Append function returns the record number of the new record, or 0 if the file could not be locked.
- *SQL*: The Append function returns the record ID.

Example

The following example demonstrates how to add a contact record in GoldMine via DDE.

```
Sub Main()
Dim sQ
Dim sWorkArea As String
Dim lChannel As Long
Dim sRet As String
sQ = Chr(34)
'Open a DDE channel
lChannel = DDEInitiate("GoldMine", "Data")
sWorkArea = DDERequest(lChannel, "[Open(Contact1)]")
If sWorkArea <> "0" Then 'Database was opened
'Append a new record to Contact1
sRet = DDERequest(lChannel, "[Append(" + sWorkArea + ")]")
If sRet <> "0" Then 'Record was Appended
StatusBar = "New Record Added"
'Replace Company name with "New Record"
sRet = DDERequest(lChannel, "[Replace(" + sWorkArea + "," + sQ(34) +
"Company" + sQ(34) + "," + sQ + "NewRecord" + sQ + ")]")
If sRet = "1" Then
StatusBar = "Replaced complete"
Else
StatusBar = "Replaced Failed"
End If
'Unlock and Close the record
sRet = DDERequest(lChannel, "[Unlock(" + sWorkArea + ")]")
```

```
sRet = DDERequest(lChannel, "[Close(" + sWorkArea + ")]")
Else
StatusBar = "Error Opening Contact1"
End If
End If
'Terminate the DDE Channel
DDETerminate (lChannel)
End Sub
```

Closing an Opened File

Syntax	[CLOSE(<work area>)]

The Close function is used to release a previously OPENed file when processing is complete. When access is complete, a file must be CLOSEd to release memory used by GoldMine to maintain database work areas.

Parameters

The Close function accepts one parameter—the work area handle of the file to close. The Open file returns the work area handle when the file is opened.

Return Value

The Close value returns 1 if the function was able to successfully close the work area, 0 if an invalid work area handle was passed.

Example

See Adding an Empty Record .

Deleting the Current Record

Syntax	[Delete(<work area>)]

The Delete function deletes the current record in the specified work area. The record pointer is not advanced to the next record.

Parameters

The Delete function takes one parameter—the work area value obtained from the Open function.

Example

```
DDERequest(lChannel, "[Delete(" + sWorkArea + ")]")
```

Creating a Subset of Records

Syntax	[FILTER(<work area>,<expression>)]

The Filter function limits access to data in a GoldMine database by creating a subset of records based on expression criteria.

Parameters

The Filter function takes two parameters. Enclose each parameter in quotation marks (“).

The first parameter is the work area handle of the file that you want to read. The Open function provides this value when the data file is opened.

The second parameter is a valid Xbase expression.

> **TIP:** To remove the filter from the database, use a Filter function with an empty string, such as [FILTER(<work area>,"")].

Example

This example will scan the current contact’s history for all activities completed by a specific user. It works by first setting the Range of history to a specific contact via the AccountNo. Once the range is set, the Filter is applied to “see” only records for a specific user within that range.

```
Sub Main()
Dim lChannel As Long
Dim sRet As String
Dim sWorkArea As String
Dim sQ As String
Dim sAccNo As String
Dim sUser As String
Dim bEOF As Boolean
Dim Counter As Integer

'Initialize some variables
Counter = 0
sQ = Chr(34)

'Get user input
sUser = InputBox("Enter a GoldMine username below.")
'Uppercase and pad the username
sUser = UCase(Left$(sUser + " ", 8))
'Start DDE Conversation with GoldMine
lChannel = DDEInitiate("GoldMine", "Data")
'Get the current AccountNo
sAccNo = DDERequest(lChannel, "Contact1->AccountNo")
'Open the ContHist file
sWorkArea = DDERequest(lChannel, "[Open(CONTHIST)]")
'If WorkArea is valid then do our thing
If sWorkArea <> "0" Then
'Set the hi/lo range to the AccountNo
sRet = DDERequest(lChannel, "[Range(" + sQ + sWorkArea + sQ + "," + sQ +
sAccNo + sQ + "," + sQ + sAccNo + sQ + ", 33)]")
'Set the filter to only return matches where user is a match
sRet = DDERequest(lChannel, "[Filter(" + sQ + sWorkArea + sQ + "," + sQ +
"USERID='" + sUser + "'" + sQ + ")]")
'Go to the Top record
sRet = DDERequest(lChannel, "[Move(" + sQ + sWorkArea + sQ + ", TOP)]")
'Determine if we have at least one match
```

```
If sRet <> "1" Then 'no matches
bEOF = True
Else 'We have at least one match
Do
'Increment the counter
Counter = Counter + 1
'Go to the next record
sRet = DDERequest(lChannel, "[Move(" + sQ + sWorkArea + sQ + ", SKIP)]")
'Determine if we have run out of matching records
If sRet <> "1" Then bEOF = True
Loop Until bEOF = True 'Loop until no more matching records
End If
'Close WorkArea
sRet = DDERequest(lChannel, "[Close(" + sQ + sWorkArea + sQ + ")]")
'Display results
MsgBox (Str$(Counter) + " history records for this contact have a User = '"
+ sUser + "'")
End If
'Close DDE channel
DDETerminate (lChannel)
End Sub
```

Checking for an Xbase or SQL Table

Syntax	[IsSQL (<work area>)]

The IsSQL function returns the table type (Xbase or SQL) that is open in a work area. Using this DDE command, you can determine the most appropriate method to retrieve information when working with DataStream—see Returning GoldMine Record Data . For example, when your routine starts, you can open Contact1 and Cal, issue an IsSQL command to determine the GoldDir and CommonDir database types, and then close both work areas. You can then send the appropriate DataStream calls.

Parameters

The IsSQL function takes work area as the only parameter.

Return Values

IsSQL returns 1 for an SQL database table, or 0 for an Xbase file.

Moving to a Specified Record

Syntax	[MOVE(<work area>,<subfunction>,<scope>)]

The Move function will position the record pointer to a particular record in a data file. Before using Move, you must open a data file using the Open function.

Parameters

The Move function requires either two or three parameters.

The first parameter is the work area handle of the file whose record pointer you want to position. The Open function provides this value when the data file is opened.

The second parameter is the name of the Move subfunction that you want to perform.

Depending on the subfunction, a third parameter can be required. The following table lists the Move subfunctions and the requirements for the third parameter:

Valid Move Subfunctions

Subfunction	Description	3rd Parameter
TOP	Move to first logical record	Not required
BOTTOM	Move to last logical record	Not required
SKIP	Skip records	Optional, records to skip
GOTO	Go to a specific record	Record number (Xbase), Record ID (SQL)
SEEK	Seek a specific record by key	Search key value
SETORDER	Select an index	Index name

Top	Positions the record pointer at the first logical record according to the current index order. For example, if the data file open in the selected work area is CONTACT1.DBF, and the index order is set to **Company**, a call to TOP will result in the record pointer being positioned at a record with a company name, such as AAA Cleaners.
Bottom	Positions the record pointer at the last logical record according to the current index order. For example, if the data file open in the selected work area is CONTACT1.DBF, and the index order is set to **Company**, a call to BOTTOM will result in the record pointer being positioned at a record with a company name, such as Z-best Bakery.
Skip	Moves the record pointer record by record. If SKIP is called without the third parameter, it will move the record pointer to the next logical record according to the current index order. If SKIP is called with a string numeric as the third parameter, the record pointer will be moved forward by the indicated number if the value is positive, or backward if the value is negative. Negative numbers must be passed in quotation marks, for example “-1”.
Goto	Positions the record pointer at the record number (Xbase) or record ID (SQL) specified by a string numeric passed as the third parameter.
Seek	Attempts to locate a record in the data file with an index key that matches the string passed as the third parameter. Partial key searches are allowed; GoldMine will position the record pointer at the record with the key that most closely matches the passed value.
Setorder	Selects an active index for ordering and seeking the data file. See SQL Database Structures for the appropriate values and collating sequence for each data file index.

TIP: If an invalid index is selected for the data file, none of the MOVE subfunctions will operate properly.

Return Value

The Move function can return several values.

Move Return Values

Return	Description
0	Error occurred
1	Record pointer successfully moved, or index selected
2	Exact match not found, pointer positioned at closest match
3	Record pointer positioned at end-of-file (EOF)
4	Record pointer positioned at beginning-of-file (BOF)

An error can be returned under any of the following conditions:

- Invalid work area handle is passed to the function.
- Invalid subfunction is passed.
- Out-of-range record number is passed.
- Nonnumeric value is passed as a third parameter when a numeric value is expected.

Example

The following example will open Contact1, perform various Move operations, and display the resulting contact name between Moves.

Note that the example below is written in Visual Basic for Applications, and the DDEInitiate and DDERequest functions are not a part of Visual Basic 6.0. DDE functionality is performed via the LinkRequest method in a textbox. For more information, see Establishing a DDE Conversation.

```
Sub Main()
Dim lChannel As Long
Dim sWorkArea As String
Dim sRet As String
Dim iX As Integer
Dim sSeekVal As String
Dim sQ As String

sQ = Chr(34)
lChannel = DDEInitiate("GoldMine", "Data")
sWorkArea = DDERequest(lChannel, "[Open(Contact1)]")

'Goto Top of Database
sRet = DDERequest(lChannel, "[Move(" + sWorkArea + ",Top)]")
MsgBox ("Top: Contact=" + DDERequest(lChannel, "[Read(" + sWorkArea + ",
Contact)]"))
'Skip forward 1 record
```

```
sRet = DDERequest(lChannel, "[Move(" + sWorkArea + ", SKIP)]")
MsgBox ("SKIP: Contact=" + DDERequest(lChannel, "[Read(" + sWorkArea + ",
Contact)]"))

'Skip X record (x=5)
iX = 5
sRet = DDERequest(lChannel, "[Move(" + sWorkArea + ",SKIP," + Str(iX) +
")]")
MsgBox ("Skip 5: Contact=" + DDERequest(lChannel, "[Read(" + sWorkArea + ",
Contact)]"))

'Goto Bottom of Database
sRet = DDERequest(lChannel, "[Move(" + sWorkArea + ", Bottom)]")
MsgBox ("Bottom: Contact=" + DDERequest(lChannel, "[Read(" + sWorkArea + ",
Contact)]"))

'Skip back 1 record (Note: the -1 must be enclosed in quotes)
sRet = DDERequest(lChannel, "[Move(" + sWorkArea + ", Skip, " + sQ + "-1" +
sQ + ")]")
MsgBox ("Skip -1: Contact=" + DDERequest(lChannel, "[Read(" + sWorkArea +
", Contact)]"))

'Goto Record 10
sRet = DDERequest(lChannel, "[Move(" + sWorkArea + ", Goto, 10)]")
MsgBox ("Goto: Contact=" + DDERequest(lChannel, "[Read(" + sWorkArea + ",
Contact)]"))

'Seek for a Company
sRet = DDERequest(lChannel, "[Move(" + sWorkArea + ",SetOrder, 16)]")
sSeekVal = UCase(InputBox("Enter a Company to search for"))
sRet = DDERequest(lChannel, "[Move(" + sWorkArea + ",Top)]")
sRet = DDERequest(lChannel, "[Move(" + sWorkArea + ", Seek, " + sQ +
sSeekVal + sQ + ")]")
MsgBox ("Seek: Contact=" + DDERequest(lChannel, "[Read(" + sWorkArea + ",
Contact)]"))

ret = DDERequest(lChannel, "[Close(" + sWorkArea + ")]")
DDETerminate (lChannel)
End Sub
```

Opening a Data File

Syntax	[OPEN(<tablename>)]

The Open function is used to open a GoldMine data file for processing by another application. This function must be called before calling any GoldMine DDE functions that work with an individual data file. It is not necessary to use this function when calling the RecordObj function, because this function opens the necessary data files automatically.

Parameters

The Open function takes one parameter—the name of the file to open. The following values are valid for this parameter:

Open Valid Parameters

File	Description
CAL	Calendar activities file
CONTACT1	Primary contact information file
CONTACT2	Primary contact information file
CONTGRPS	Groups file
CONTHIST	History records file
CONTSUPP	Supplementary records file
INFOMINE	InfoCenter file
LOOKUP	Lookup file
MAILBOX	E-mail Center mailbox file
OPMGR	Opportunity Manager file
PERPHONE	Personal Rolodex file
RESOURCE	Resources file
SPFILES	Contact files directory

Return Value

The Open function returns an integer value representing the handle to the file's work area. This value is required for all subsequent access to the file. If the file could not be opened, or an invalid parameter is passed, the function will return 0.

Example

See Adding an Empty Record .

Limiting GoldMine Search Range

Syntax	[RANGE(<work area>,<minimum>,<maximum>,<tag>)]

The Range function activates the index in a table and sets a range of values to limit the scope of data that GoldMine will search.

Parameters

The Range function requires four parameters.

The first parameter is the work area handle of the file that you want to read. The Open function provides this value when the data file is opened.

The second parameter is the minimum value of the range. Enclose this parameter in quotation marks (“).

The third value is the maximum value of the range. Enclose this parameter in quotation marks (“).

The fourth value is the tag that corresponds to the index file. For details about tags, see SQL Database Structures.

Example

See Creating a Subset of Records

Reading a Field Value

Syntax	[READ(<work area>,<field>)]

The Read function is used to query a data file for the value of a field. Before using Read, you must open a data file using the Open function. In addition, you will probably want to position the record pointer to the record you want to query by using the Move function.

Parameters

The Read function requires two parameters.

The first parameter is the work area handle of the file that you want to read. The Open function provides this value when the data file is opened.

The second parameter is the name of the field in the data file whose value you want to query. You will normally pass only a single field name, such as CONTACT as the second parameter. However, if you pass a field expression, such as “*COMPANY + CONTACT*” GoldMine will attempt to evaluate the expression and return the value of the expression. When an expression is passed as the second parameter, the expression must be surrounded by quotation marks.

Return Value

The Read function returns a character string containing the value in the specified field, or the value of the specified expression. If an error occurs, the Read function returns a null string. The error could be caused by an invalid work area handle, an invalid field being passed, or an expression that GoldMine could not evaluate.

Example

See Moving to a Specified Record

Checking the Current Record Number or Record ID

Syntax	[RECNO(<work area>)]

- *Xbase*: RecNo function is used to determine current record number position.
- *SQL*: RecNo function is used to determine the record ID.

Parameters

The RecNo function accepts one parameter—the work area handle of the file. The work area handle is returned by the Open file when the file is opened.

Return Value

The RecNo function returns the current record number position, 0 if an invalid work area handle was passed.

Example

The following example will get the current Contact1 RecNo and display it in the GoldMine status bar.

Note that the example below is written in Visual Basic for Applications, and the DDEInitiate and DDERequest functions are not a part of Visual Basic 6.0. DDE functionality is performed via the LinkRequest method in a textbox. For more information, see Establishing a DDE Conversation.

```
Sub Main()
Dim lChannel As Long
Dim sWorkArea As String
Dim sRet As String
Dim sRecNo As String
Dim sQ As String

sQ = Chr(34)
lChannel = DDEInitiate("GoldMine", "data")
sWorkArea = DDERequest(lChannel, "[Open(Contact1)]")
sRecNo = DDERequest(lChannel, "[RecNo(" + sWorkArea + ")]")
sRet = DDERequest(lChannel, "[Close(" + sWorkArea + ")]")
sRet = DDERequest(lChannel, "[StatusMsg(" + sQ + "RecNo=" + sRecNo + sQ +
")]")
MsgBox ("GoldMine's status bar should now display the RecNo ")
End Sub
```

Changing a Field Value

Syntax	[REPLACE(<work area>,<field>,<value>,<append>)]

The Replace function is used to change the value in a particular field in one GoldMine data file. Before using Replace, you must open a data file using the Open function. In addition, you will probably want to position the record pointer to the record you want to change either by using the Move function, or by adding a new record with the Append function.

After executing the Replace function, GoldMine will update the specified field with the new value, and update the appropriate remote synchronization data structures to indicate that the field was changed.

In a network environment, GoldMine automatically locks the record before performing the replacement. The record is not automatically unlocked, allowing for fast multiple field replacements. The record is automatically unlocked when a Close, Move, or Unlock command is issued on the work area.

Parameters

The Replace function requires three parameters and has an optional fourth parameter.

The first parameter is the work area handle of the file in which you want to perform the replacement. The Open function provides this value when the data file is opened.

The second parameter is the name of the field to be replaced. See SQL Database Structures for information on the name of fields in each GoldMine data files. If you attempt to replace a field that does not exist in the file open in the specified work area, the Replace function will fail.

The third parameter is the value to replace. This value must be enclosed in quotation marks. The replace value must be a string value. If the replacement field is a date or numeric field, GoldMine will convert the string data to the appropriate data type prior to performing the replacement.

The fourth parameter will add data instead of replacing data. Using this parameter, you can insert large amount of text into a notes field. To append instead of replace incoming data from the third parameter, pass 1 as the fourth parameter. You can set up a loop to feed notes in 256-byte segments to override the 256-byte limit for inbound DDE requests.

Return Value

If the file was replaced, the Replace function returns 1. If the field could not be replaced, 0 is returned. The failure can be caused under any of the following conditions:

- Invalid parameter, such as an invalid work area handle.
- Invalid field name.
- Record already locked by another user.

Example

See Adding an Empty Record .

Performing a Sequential Search

Syntax	[SEARCH(<work area>,<expression>,<index>)]

The Search function is used to perform a sequential search on a file. Unlike Move, Search scans the table, one record at a time, looking for a record that satisfies the search condition. The search condition can be any Xbase expression that GoldMine understands, but is usually an expression that tests the value of one or more fields in the file. When a match is found, the record pointer is located at the matching record.

Search starts with the record that immediately follows the current record (the next logical record according to the selected index order) and continues until a match is found or the end of file is encountered. Because of this, Search can be called repeatedly to return a list of records that satisfy the search condition.

Parameters

The Search function takes three parameters.

The first parameter is the work area handle of the file you want to search. The Open function provides this value when the data file is opened.

The second parameter is the search expression, such as "CITY='Los Angeles'"
The expression must be surrounded by quotation marks, and any string literal characters with the expression must be surrounded by single quotes (').

The third parameter is the optional index order to use when searching the data file. When this parameter is not specified, the data file is searched by record number (physical) order. See Xbase Database Structures for the appropriate values and collating sequence for each data file's indexes.

NOTE: If an invalid index is selected for the data file, the Search function will not operate properly.

Return Value

The Search function can return several values.

Search Return Values

Return	Description
0	Error occurred or match could not be found
>1	Match found; return value indicated current physical record number (Xbase) or record ID (SQL)

An error can be returned if an invalid work area handle is passed to the function, or if an invalid search condition is passed.

Example

The following example will prompt the user for a city name, then display the contact name for the first matching record.

NOTE: The example below is written in Visual Basic for Applications, and the DDEInitiate and DDERequest functions are not a part of Visual Basic 6.0. DDE functionality is performed via the LinkRequest method in a textbox. For more information, see Establishing a DDE Conversation.

```
Sub Main()
Dim lChannel As Long
Dim sWorkArea As String
Dim sRet As String
Dim sSeekVal As String
Dim sQ As String
sQ = Chr(34)
lChannel = DDEInitiate("GoldMine", "Data")
sWorkArea = DDERequest(lChannel, "[Open(Contact1)]")

'Search for a City
sSeekVal = UCase(InputBox("Enter a City to search for"))
sRet = DDERequest(lChannel, "[Move(" + sWorkArea + ",Top)]")
sRet = DDERequest(lChannel, "[Search(" + sWorkArea + "," + sQ + "Upper
(CITY)='" + sSeekVal + "'" + sQ + ")]")
If sRet = "" Then
MsgBox ("Search: No Match")
Else
MsgBox ("Search: Contact=" + DDERequest(lChannel, "[Read(" + sWorkArea + ",
Contact)]"))
End If
ret = DDERequest(lChannel, "[Close(" + sWorkArea + ")]")
DDETerminate (lChannel)
End Sub
```

Unlocking a Record

Syntax	[UNLOCK(<work area>)]

The Unlock function unlocks a record previously locked by a call to either Append or Replace. GoldMine does not specifically release a lock on a record until you call Unlock, allowing you to perform multiple field replacements quickly. Before using Unlock, you must open a data file using the Open function.

After calling Unlock, GoldMine will also update the remote synchronization data structures to indicate the date and time that the record was modified.

Parameters

The Unlock function accepts one parameter—the work area handle of the file to close. The work area handle is returned by the Open file when the file is opened.

Return Value

The Unlock function returns 1 if the record was unlocked, or 0 if an invalid work area handle was passed to the function.

Example

See Adding an Empty Record

Accessing Contact Records

For specific applications that need access to the GoldMine contact database at the logical level, the RecordObj function is the preferred access method. Unlike the low-level DDE functions, the RecordObj function maintains all of the relationships between the various GoldMine files. This access method is most often used for document merging functions such as word processor mail merges or placing information into a spreadsheet.

Linking GoldMine Fields with an External Application

Syntax	[RECORDOBJ(<subfunction>,<scope>)]

The RecordObj function is a specialized function designed to link DDE fields in a document application, such as a word processor or spreadsheet. Using RecordObj, an application can access the contact record in a high-level fashion, rather than opening the CONTACT1.DBF and CONTACT2.DBF files using Open.

Calling RecordObj within a DDE program is equivalent to viewing and manipulating the contact record within GoldMine. The calling program can control the record pointer in the contact record much the same way a GoldMine user can move the record pointer. In fact, RecordObj can be called in such a way as to create a minimized contact record in the GoldMine work area display.

The primary differences between using Open, Move, and Read to access contact information and using RecordObj are described in the following table.

Differences in Accessing Contact Information

Using Open, Move, Read	Using RecordObj
Any filter or group that is active on a contact record in GoldMine is ignored when files are accessed using Open and Move	RecordObj can work in conjunction with a filter or group. Any records that do not match the filter expression, or are not members of the group, are skipped
The only way to maintain the relationship between the CONTACT1 and CONTACT2 files, is to manually reposition CONTACT2 whenever the record pointer is moved in CONTACT1.DBF.	Automatically maintains the relationship between CONTACT1 and CONTACT2, and other contact information such as history.
	RecordObj does not contain a method to read specific fields from the database. It is expected that the application will use DDE link fields or the Expr function to query information from the database, and use RecordObj function calls only to position the record pointer.
	When RecordObj is used to move the record pointer, the contact record screen in GoldMine is updated, and a DDE Warm Link Advise message is sent to all DDE link fields, automatically updating these fields with the new contact information.

Parameters

The RecordObj function requires either one or two parameters.

The first parameter is the name of the RecordObj subfunction that you want to perform.

Depending on the subfunction, a second parameter can be required. The following table lists the RecordObj subfunctions and the requirements of the second parameter.

Valid RecordObj Functions

Subfunction	Description	2nd Parameter
SETOBJECT	Create or select contact record	Optional object pointer
TOP	Move to first logical record	Not required
BOTTOM	Move to last logical record	Not required
SKIP	Skip records	Optional, recs to skip
SEEK	Seek a specific record by key	Search key value
SETORDER	Select an index	Index tag number
GETORDER	Return the currently active index name	Not required
SETTITLE	Set the contact record title	Text of title

CLOSEWINDOW	Close the contact record	None
SETRECORD	Change the behavior of SKIP, TOP, and bottom	Name of data structure to be queried
REFRESH	Repaint the contact record	Not required
GETRP	Return the point to the current contact record (Xbase) or the record ID (SQL)	Not required
GETFILTEREXPR	Get the activated filter's expression	Not required
GETGROUPNO	Get the GroupNo of the activated group	Not required

Setobject	The SetObject call must be called prior to calling any other RecordObj subfunction to specify the contact record that subsequent RecordObj calls will manipulate.If SetObject is called without a second parameter, subsequent calls to RecordObj will manipulate the currently active contact record. The user can change the active contact record in GoldMine while the DDE conversation is active, but this will not affect the contact record that is linked to the RecordObj function.If SetObject is called with a second parameter of 0, GoldMine will create a minimized contact record in the work area display, and subsequent calls to RecordObj will manipulate that contact record. If SetObject is called with a second parameter of 1, GoldMine will create a minimized contact record in the work area display and copy any filter or group active on the last used contact record into the newly minimized contact record.If RecordObj is called with a specific pointer number, GoldMine will attempt to establish a link with that contact record. A client application can obtain this pointer only when using the GoldMine document merging feature, when GoldMine, acting as a DDE client, passes this long pointer as the seventh parameter.
Top	Positions the record pointer at the first logical record according to the current index order. For example, if the contact record index order is set to *Company*, a call to Top will result in the record pointer being positioned at a record with a company name such as "AAA Cleaners." GoldMine will also update the contact record to display the new record.
Bottom	Positions the record pointer at the last logical record according to the current index order. For example, if the contact record index order is set to *Company*, a call to Bottom will result in the record pointer being positioned at a record with a company name such as "Z-best Bakery." GoldMine will also display the new record.

Skip	The Skip subfunction moves the record pointer on a record-by-record basis.If Skip is called without the second parameter, it will move the record pointer to the next logical record according to the current index order.If Skip is called with a string numeric as the second parameter, the record pointer will be moved forward by the indicated number of records if the value is positive, or backwards if the value is negative. Negative numbers must be passed in quotation marks, for example "-1." GoldMine will also update the display to show the new record.The Skip subfunction is sensitive to any filter or group that can be active on the contact record in GoldMine. For example, if the user applies a filter to the contact record in GoldMine, the Skip subfunction will skip over any records that do not match the filter expression.
Goto	The Goto subfunction positions the record pointer at the record number specified by a string numeric passed as the second parameter.
Seek	Attempts to locate a record in the data file with an index key that matches the string passed as the second parameter. Partial key searches are allowed, and GoldMine will position the record pointer at the record with the key that most closely matches the passed value. GoldMine will update the display to show the new record.
Setorder	Selects an active index for ordering and SEEKing the contact database. Only the twelve CONTACT1 indexes can be used for this subfunction. See Xbase Database Structures for the appropriate values and collating sequence for each data file's indexes.
Getorder	Returns the active index being used to sort the contact records. See Xbase Database Structures for the appropriate values and collating sequence for each data file's indexes.
Settitle	Changes both the text in the title bar of the contact record's window and the text displayed below a minimized contact record. For example, a DDE application that merges contact records within a document can modify the contact record title to indicate the number of records that have been merged. Any text that is passed as the second parameter must be enclosed in quotation marks, and will be used as the new title's text.
Closewindow	Closes the contact record when processing is complete. Issuing this call is equivalent to selecting *Close* from the contact record's system menu.
Setrecord	Changes the behavior of the Skip, Top, and Bottom subfunctions to allow ancillary contact information (such as additional contacts) to be queried using the RecordObj function. Normally, GoldMine assumes the CONTACT1 data file to be the parent data file, and when the Skip, Top, or Bottom subfunction is called, the record pointer is repositioned in this data file. When accessing information in GoldMine tabs, however, the Skip, Top, and Bottom subfunctions must be able to reposition the record pointer in the data file that stores these items (CONTSUPP).The SetRecord subfunction accepts the name of the data structure being queried as the second parameter. Valid data structure names are listed in the following table.

Setrecord Valid Structure Names

Data Structure Name	Description
CONTACTS	Additional contacts
PROFILE	Profile records
REFERRALS	Referral records
LINKS	Linked documents
PRIMARY	Primary contacts

Using SetRecord changes the behavior of the Skip, Top, and Bottom subfunctions.

The first parameter is the name of the RecordObj subfunction that you want to perform. When Top is called, GoldMine will position the record pointer in the supplementary data file so that the first record containing the selected information is the current record. For example, if SetRecord is used to select CONTACTS, Top will position the record pointer on the first additional contact record for the current contact. The record pointer in the primary information data file (CONTACT1) will not be moved, so the name of the current company will remain the same. Bottom behaves in a similar manner.

Skip will position the record pointer in the supplementary file on the next record of the selected type. For example, if SetRecord is used to select CONTACTS, Skip will position the record pointer in the supplementary file on the next additional contact record for the current contact. The record pointer in the primary information data file (CONTACT1) will not be moved, unless the record pointer in the supplementary file was already positioned at the last record of the selected type; then GoldMine will reposition the record pointer in the primary information data file (CONTACT1) to the next contact record and reset the record pointer in the supplementary file to the first supplemental record of the selected type. DDE macros are also sensitive to the setting of the SetRecord subfunction (see DDE Macros.)

Refresh	Repaints the contact record
GetRP	Obtains a pointer of the currently selected contact record
GetGroupNo	Returns the group number (if a group is activated)
GetFilterExpr	Returns the filter expression (if a filter is activated)

Return Value

All RecordObj subfunctions return 1 if the function was completed successfully or 0 if an internal error occurred.

Example

The following example will count the number of additional contacts for the current contact.

Note that the example below is written in Visual Basic for Applications, and the DDEInitiate and DDERequest functions are not a part of Visual Basic 6.0. DDE functionality is performed via the LinkRequest method in a textbox. For more information, see Establishing a DDE Conversation.

```
Sub Main()
Dim lChannel As Long
Dim sAccountNo As String
```

```
Dim sRet As String
Dim sANRT As String
Dim iAddCount As Integer

lChannel = DDEInitiate("GoldMine", "Data")
sAccountNo = DDERequest(lChannel, "Contact1->AccountNo")
sRet = DDERequest(lChannel, "[RecordObj(SetObject, 1)]")
sRet = DDERequest(lChannel, "[RecordObj(SetRecord, Contacts)]")
sRet = DDERequest(lChannel, "[RecordObj(Top)]")
sANRT = DDERequest(lChannel, "Trim(ContSupp->AccountNo)+Trim(ContSupp-
>RecType)")
iAddCount = 0
While sANRT = sAccountNo + "C"
iAddCount = iAddCount + 1
sRet = DDERequest(lChannel, "[RecordObj(Skip)]")
sANRT = DDERequest(lChannel, "Trim(ContSupp->AccountNo)+Trim(ContSupp-
>RecType)")
Wend
sRet = DDERequest(lChannel, "[RecordObj(CloseWindow)]")
MsgBox (Str(iAddCount) + " Additional Contacts")
DDETerminate (lChannel)
End Sub
```

Accessing Specialized DDE Functions

GoldMine provides a set of specialized functions for performing specific tasks, such as adding document links to the contact database or sending GoldMine a CallerID message.

Retrieving Login Credentials for Use with the GMXS32.DLL

Syntax	[GetLoginCredentials]

GoldMine Version 5.70.20222

The GetLoginCredentials function is used to retrieve a string containing login credentials to be used for logging into the `GMXS32.DLL` through the GMW_LoadAPI, GMW_LoadBDE or GMW_Login functions. Using this option, it is not necessary to prompt the integration user for login information if GoldMine is running. The login credentials received are only valid for 30 seconds, so do not store them and attempt to use them at a later time. The string returned by this command should be used as the password to the appropriate login function, where the username is "**DDE_LOGIN_CREDENTIALS**".

Example

This example retrieves various parameters from GoldMine and passes them to the GMW_LoadAPI or GMW_LoadBDE function in the `GMXS32.DLL`.

The following example is written in Visual Basic 6.0 using the DDEInitiate and DDERequest functions defined in Establishing a DDE Conversation.

```
With frmDDE
 iResult = .DDEInitiate
```

```
 If iResult Then
 frmPaths.txtSysFolder = .DDERequest("&SysDir")
 frmPaths.txtGoldDir = .DDERequest("&GoldDir")
 frmPaths.txtCommonDir = .DDERequest("&CommonDir")
 sLoginCredentials = .DDERequest("[GetLoginCredentials]")
lResult = GMW_LoadBDE(frmPaths.txtSysFolder, frmPaths.txtGoldDir, _
 frmPaths.txtCommonDir, "*DDE_LOGIN_CREDENTIALS*", _
 sLoginCredentials)
End With
```

Retrieving the RecID of the Current Opportunity

Syntax	[GetActiveOppty]

GoldMine Version 5.70.20222

The GetActiveOppty function is used to retrieve the RecID of the currently selected Opportunity in the Opportunity Manager.

Return Value

The GetActiveOppty function returns the record ID of the currently selected opportunity. If no opportunity is available, an empty string is returned.

Example

The following example reads the currently selected Opportunity's record ID and displays the value in a message box.

The following example is written in Visual Basic 6.0 using the DDEInitiate and DDERequest functions defined in Establishing a DDE Conversation.

```
With frmDDE
 iResult = .DDEInitiate
 If iResult Then
 sResult = .DDERequest("[GetActiveOppty]")
 MsgBox sResult
 End If
End With
```

Completing a Calendar Activity

Syntax	[CalComplete(<RecNo>,<ActvCode>,<ResultCode>, <User>,<Ref>,<Notes>,<RetainDate>)]

The CalComplete function is used to complete an activity from the Calendar.

Parameters

The CalComplete function takes up to seven parameters. All parameters must be passed in quotation marks.

The first parameter is the record number of the calendar activity to be completed.

The second parameter is the Activity Code. This parameter is optional.

The third parameter is the Result Code. This parameter is optional.

The fourth parameter is the User. If this parameter is not specified, the User field defaults to the currently logged user.

The fifth parameter is the history Reference. This parameter is optional.

The sixth parameter is the Notes for the history record. This parameter is optional.

The seventh parameter indicates whether the function should retain its original date, or use the current date/time. To retain the original date, set this value to 1.

Return Value

The CalComplete function returns the record number (Xbase) or record ID (SQL) of the new history record created.

Example

This example will open the CAL file, read the current RecNo (Xbase), or RecID (SQL), and complete the record to History.

NOTE: The example below is written in Visual Basic for Applications, and the DDEInitiate and DDERequest functions are not a part of Visual Basic 6.0. DDE functionality is performed via the LinkRequest method in a textbox. For more information, see Establishing a DDE Conversation.

```
Sub Main()
Dim lChannel As Long
Dim sRet As String
Dim sRecNo As String
Dim sHRecNo As String
Dim sWorkArea As String
Dim sQ As String

sQ = Chr(34)
lChannel = DDEInitiate("GoldMine", "Data")

sWorkArea = DDERequest(lChannel, "[Open(CAL)]")
sRecNo = DDERequest(lChannel, "[RecNo(" + sQ + sWorkArea + sQ + ")]")
sHRecNo = DDERequest(lChannel, "[CalComplete(" + sQ + sRecNo + sQ + ")]")
MsgBox ("New History Record Number: " + sHRecNo)
DDETerminate (lChannel)
End Sub
```

Displaying the Contact Record of an Incoming Caller

Syntax	[CALLERID(<telephone>,<message>,<display dialog>)]
	[CallerIDAll(<phone>, <message>, <displayDlg>, <bUPhone>)]

The CallerID and CallerIDAll functions are used to inform the GoldMine user that an incoming call has been identified by Automatic Number Identification (ANI) equipment attached to the telephone system. By using the caller ID functions, GoldMine can perform a lookup on the contact database, and attempt to locate a contact record with a telephone number that matches the telephone number extracted by the ANI device.

With the caller ID functions, GoldMine can automatically display the contact record of the caller. A dialog box is displayed, allowing the user to select an action. A CallerID function parameter is used to specify the message in the dialog box.

The two functions perform the same functionality with the difference of the CallerIDAll command will search all phone numbers for the contact record (except FAX), instead of just the Phone1 field.

Parameters

The caller ID functions accept three parameters. The CallerIDAll function accepts a fourth parameter that the CallerID function does not:

The first parameter is the telephone number of the caller as captured by the ANI device. The calling application is responsible for formatting the telephone number that appears in the Phone1 field in GoldMine. Enclose this parameter in quotation marks (“).

The second parameter is the optional message to be displayed in the dialog box in GoldMine. Enclose this parameter in quotation marks (“).

The third parameter specifies whether the dialog box is displayed. This parameter is the sum of the required options. For example, to display the caller’s contact record in the current window if the record is found, or to display the contact listing if the caller’s phone number is not found, specify 6 (2+4) as the <display dialog> parameter. The following table lists valid parameter values.

CallerID Parameters

Value	Description
0	Dialog box is displayed (default when third parameter is not passed)
1	Dialog box is not displayed, and contact record is displayed in a new contact record
2	Dialog box is not displayed, and contact record is displayed in the current contact record
4	Contact Listing is displayed when GoldMine cannot find the contact’s telephone number. To activate this option, add this value to the third parameter value.
8	Restores input focus to the window that had input focus just before CALLERID is called—used by applications that control the entire interface.

The fourth parameter that is only accepted by the CallerIDAll function is whether or not to search the UPhone fields stored in Contact2. Set to 1 to search the UPhone fields, or 0 to omit the UPhone fields.

Return Values

CallerID Return Values

Return	Description
0	Error occurred
1	Contact record found

2	Contact record not found

Example

The following example demonstrates the CallerID function.

NOTE: The example below is written in Visual Basic for Applications, and the DDEInitiate and DDERequest functions are not a part of Visual Basic 6.0. DDE functionality is performed via the LinkRequest method in a textbox. For more information, see Establishing a DDE Conversation.

```
Sub Main()
 Dim lChannel As Long
 Dim sRet As String
 Dim sPhone As String
 Dim sQ As String
sQ = Chr(34)
 lChannel = DDEInitiate("GoldMine", "Data")
 sPhone = InputBox("Enter Phone to Look Up. Format:(###)###-####")
 sRet = DDERequest(lChannel, "[CallerID(" + sQ + sPhone + sQ + ")]")
End Sub
```

Running a Counter

Syntax	[COUNTER(<string>,<inc>,<start>,<action>)]

The Counter function returns a sequence of consecutive numbers each time the expression is evaluated.

Parameters

The counter name must be unique, and can be a maximum of 10 characters. Each evaluation of the Counter function increments the counter by the <inc> value.

The <start> and <action> parameters are optional. When <action> is 1, the start value resets the counter. When <action> is 2, the counter is deleted. Counter stores the count value between GoldMine sessions, and it is shared by all GoldMine users.

GoldMine can track an unlimited number of uniquely named counters. The counter values are stored in the LOOKUP table.

Return Value

The Counter function returns a number incremented by <inc>.

Example

```
[Counter("InvoiceNo",1,1000)]
```

Returning GoldMine Record Data

Syntax	[DATASTREAM(<subcommand>,<parameter>)]

DataStream returns the data of ordered records from any GoldMine table using the most efficient method possible. The caller can specify the fields and expressions to return, as well as the range of records to return. A filter can optionally be applied to the data set.

The DataStream method allows for many useful applications. One example would be to publish the contents of GoldMine data on the Internet by merging HTML templates with the data returned by DataStream. Web pages can be created to display GoldMine data requested by a visitor. Based on the visitor's selections, a company could dynamically present a variety of HTML pages, such as:

- Addresses of product dealers in a particular city
- Financial numbers stored in Contact2
- Seating availability of upcoming conferences

With a fast Internet connection and a strong SQL server, the GoldMine client could simultaneously respond to dozens of requests.

Record Selection

The DataStream command consists of four subcommands. Each subcommand takes different parameters. The subcommands are shown below, in the order in which they must be called:

[DataStream("range", sTable, sTag, sTopLimit, sBotLimit, sFields, sFilter, sFDlm, sRDlm)]
[DataStream("query", sSQL, sFilter, sFDlm, sRDlm)]
[DataStream("fetch", nRecords, iHandle)]
[DataStream("close", iHandle)]

The "range" or "query" subcommands must be called first to request the data. The "range" and "query" subcommands return an integer handle, iHandle, which must be passed to the "fetch" and "close" subcommands. You must use either "range" or "query"—not both.

[DataStream("range", sTable, sTag, sTopLimit, sBotLimit, sFields, sFilter, sFDlm, sRDlm)]

Parameters

The sTable, sTag, sTopLimit, and sBotLimit parameters determine the range of records to scan, similar to the DDE SETRANGE command. The sFields parameter specifies the requested fields and expression to return.

The sField parameter passed to the "range" subcommand should consist of the field names and Xbase expressions to evaluate against each record in the data set. Each field must be terminated with the semicolon (;) character. Xbase expressions must be prefixed with the ampersand (&) character and terminated with a semicolon.

The other "range" parameters are optional.

Return Value

The "range" subcommand returns a range of records based on an index.

```
[DataStream("query", sSQL, sFilter, sFDlm, sRDlm)]
```

The "query" subcommand sends the sSQL query for evaluation on the server.

Parameters

The SQL query can join multiple tables and return any number of fields. The optional sFilter parameter can specify a Boolean Xbase filter expression to apply to the data set (even on SQL tables), similar to the DDE SETFILTER command. The optional sFDlm and sRDlm parameters can override the return packet's default field and record delimiters of CR and LF.

```
[DataStream("fetch", nRecords, iHandle)]
```

The "fetch" subcommand returns a single packet string that contains the requested data from all records processed by the current "fetch" command, as specified by the second nRecords parameter. iHandle must be the value returned from "range" or "query." The "fetch" command can be issued multiple times, with positive and negative values, to scroll down or up the cursor. See "Return Packet" below.

```
[DataStream("close", iHandle)]
```

The "close" subcommand must be called when the operation is complete. Unclosed data streams will leak memory and leave the database connections needlessly open. Passing an iHandle of 0 closes all open DataStream objects (of all DDE conversations).

Example 1

The following commands request the first 100 cities from the Lookup file, including the city name and record number (RecID under SQL):

```
[DataStream("range", "lookup", "lookup", "CITY", "CITYZ", "Entry; &RecNo
();")]
 [DataStream("fetch", 100, iHandle)]
 [DataStream("close", iHandle)]
```

Example 2

The following commands request the first 10 profiles of the current contact record, followed by a request for the next 50:

```
[DataStream("range","contsupp","contspfd", sAccNo+"P", sAccNo+"P",
"Contact;ContSupRef;")]
 [DataStream("fetch", 10, iHandle)]
 [DataStream("fetch", 50, iHandle)]
 [DataStream("close", iHandle)]
```

Return Packet

The "fetch" command returns a single packet string containing the data from all requested records. The packet includes a header record, followed by one record for each record evaluated by "fetch." Within each record in the packet, the fields are separated by a Field Delimiter, the carriage return character by default (13 or 0x0D). The records in the packet are separated by the Record Delimiter, the line feed character by default (10 or 0x0A). These delimiters are convenient when the requested data does not contain notes from blob fields. Otherwise, you must override the default delimiters by passing other delimiter values to the "range" and "query" commands. The characters 1 and 2 would probably make good delimiters for packets with notes.

The City Lookup example from above might return a packet of data similar to:

```
3000-0004
```

```
Boston|23
London|393
Los Angeles|633
New York|29
```

The packet header record consists of two sections. The first byte can be 0, 3 or 4. Zero indicates that more records are available, which could be fetched with another "fetch" command. A value of 3 indicates the end-of-file (EOF), and 4 indicates the beginning-of-file (BOF). The number following the dash indicates the total number of data records contained in the packet.

Packets should be designed to be 8K to 32K. DataStream takes about as much time to read three records as it does to read 30. For best performance, adjust the number to records requested by the "fetch" command to return packets of 8K to 32K.

Performance

DataStream is the fastest way to read data from GoldMine tables. Used correctly, the GoldMine DataStream will return the data faster than most development environments would directly. DataStream offers the following advantages:

1. DataStream issues a single, efficient SQL query or Xbase seek to retrieve the records from the back-end database to the local client. On SQL databases, requests of a few hundred records could be sent from the server to the client with a single network transaction, thereby minimizing network traffic.
2. All fields and expressions are parsed initially by the "range" and "query" commands, then quickly evaluated against each record in the "fetch" command. Other DDE methods (and development environments) require that each field be parsed and evaluated each time the field's data is read. This can save a significant amount of time when reading hundreds or thousands of records.
3. Only three DDE calls are required to read all the data. Using traditional record-by-record querying would require one DDE call for each field of each record (reading 10 fields from 50 records would require 500 DDE calls).
4. All the work to gather and format the data is done in fast and efficient C. The caller needs only to parse the resulting packet string.

The "range" and "query" commands execute equally fast on SQL databases. The "range" command executes much faster on Xbase tables than the "query" command.

Example 3

The following DataStream command returns all e-mail addresses in the current contact file.

```
[DataStream("range", "contsupp","contspfd","PINTERNET A","PINTERNET
B","ContSupRef;")]
[DataStream("fetch", 999, 1)]
[DataStream("close", 1)]
```

To return only the e-mail addresses of people at GoldMine, add a filter to the "range" command:

```
[DataStream("range", "contsupp","contspfd","PINTERNET A","PINTERNET AZ",
 "ContSupRef;AccountNo;&Recno();", "'@goldmine.com' $ lower(ContSupRef)")]
```

Example 4

The following DataStream returns all entries from all F2 lookups. The fields are delimited with a comma, and the records with the default LF.

```
[DataStream("range", "lookup", "lookup", "A",
"Z","FieldName;Entry;","","")]
[DataStream("fetch", 2000, 1)]
[DataStream("close", 1)]
```

Example 5

The following DataStream returns the exact packet as the one above, but using an SQL query:

```
[DataStream("query", "select fieldname, entry from lookup where fieldname >
'A' order by
fieldname, entry", "","")]
```

Processing a Web Import Instruction File

Syntax	[ExecIniImp(<filename>)]

GoldMine can send a DDE command to process a Web import instruction file. Using a DDE command allows other applications to create contact records in GoldMine. To start processing an instruction file via DDE, send the *ExecIniImp(<filename>)* command; for example, [ExecIniImp("c:\goldmine\imp.ini")].

NOTE: For details about setting up and working with the GoldMine Web Import Gateway, see "Capturing Web Data" in Maintaining GoldMine.

Reading an Xbase Expression Without Opening a File

Syntax	[EXPR(<expression>)]

The Expr function is similar to the Read function in that it attempts to evaluate an Xbase expression and return the result as a string. The Expr function, however, does not require you to open a specific data file using the Open function. The expression passed to the Expr function is evaluated against the current operating state of GoldMine (usually, the currently displayed record), rather than the state of a specific work area. For this reason, you should be aware that differences between the return values could exist for the same expression passed to Read and Expr.

Parameters

The Expr function takes one parameter—the Xbase expression to be evaluated. GoldMine supports a subset of the Xbase dialect, so there is substantial flexibility in the application of this function. Enclose this parameter in quotation marks (").

When referencing field names within an expression, you should always use an alias; otherwise, GoldMine assumes CONTACT1 to be the default alias.

Return Value

The Expr function returns a character string containing the value of the specified expression. If an error occurs, or the expression could not be evaluated, the Expr function will return a null string.

Example

The following expression will return the number of characters in notes file of the current contact.

NOTE: The example below is written in Visual Basic for Applications, and the DDEInitiate and DDERequest functions are not a part of Visual Basic 6.0. DDE functionality is performed via the LinkRequest method in a textbox. For more information, see Establishing a DDE Conversation.

```
Sub Main()
Dim lChannel As Long
Dim sExpr As String
Dim sRet As String
Dim sQ As String

sQ = Chr(34)
lChannel = DDEInitiate("GoldMine", "Data")
sExpr = "Length(Contact1->Notes)"
sRet = DDERequest(lChannel, "[EXPR(" + sQ + sExpr + sQ + ")]")
MsgBox ("Notes Length = " + sRet + " characters")
End Sub
```

Adding Merge Fields to a Form

Syntax	[FORMADDFIELDS(<FormNo>,<Fields>)]

The FormAddFields function adds merge fields to a form profile.

Parameters

The FormAddFields function takes two parameters. Enclose each parameter in quotation marks (“).

The first parameter is the number of the form.

The second parameter is a string that lists fields, macros, and expressions; each item in the string is separated by a semicolon (;). GoldMine parses the string, checks for duplication, assigns names to the fields, and then stores the items.

Example

The following example shows how to export a data file with GoldMine. It uses all of the Formxxxx functions, such as FORMADDFIELDS, FORMNEWFORM, FORMQUERYCREATE, FORMCLEARFIELDS, FORMCLOSEFORM, and FORMGETFIELDNAME.

NOTE: The example below is written in Visual Basic for Applications, and the DDEInitiate and DDERequest functions are not a part of Visual Basic 6.0. DDE functionality is performed via the LinkRequest method in a textbox. For more information, see Establishing a DDE Conversation.

```
Sub Main()
 Dim lChannel As Long
 Dim sRet As String
 Dim sFieldList As String
 Dim sFormNo As String
 Dim sFile As String
 Dim sNumRecs As String
 Dim sMergeCode As String
 Dim sQ As String
```

```
sMergeCode = ""
 sQ = Chr(34)
 'Populate the field list
 sFieldList = "&Contact ; Phone1 ; Contact1->State ; SUBSTR(Company,1,5)"
 lChannel = DDEInitiate("GoldMine", "Data")
 'Get a new Form Number
 sFormNo = DDERequest(lChannel, "[FormNewFormNo()]")
 'Register the fields
 sRet = DDERequest(lChannel, "[FormAddFields(" + sQ + sFormNo + sQ + "'" +
sQ + sFieldList + sQ + ")]")
 'Display the field names as assigned by GoldMine
MsgBox ("&Contact=" + FieldName(lChannel, sFormNo, "&Contact"))
 MsgBox ("Phone=" + FieldName(lChannel, sFormNo, "Phone1"))
 MsgBox ("Contact1->State=" + FieldName(lChannel, sFormNo, "Contact1-
>State"))
 MsgBox ("SUBSTR=" + FieldName(lChannel, sFormNo, "SUBSTR(Company,1,5)"))
'Give the output file a name
 sFile = "C:\GMDATA.DBF"
 'Create the file
 sNumRecs = DDERequest(lChannel, "[FormCreateFile(" + sQ + sFormNo + sQ +
"," + sQ + sFile + sQ + "," + sQ + "21" + sQ + ", " + sQ + sMergeCode + sQ
+ ")]")
 While DDERequest(lChannel, "[FormQueryCreate(0)]") <> "-1"
 'wait until DBF is created
 Wend
 'Clear the fields since we will not use them again
 sRet = DDERequest(lChannel, "FormClearFields(" + sQ + sFormNo + sQ + ")]")
 'Close the file when done
 sRet = DDERequest(lChannel, "FormCloseForm()")
 MsgBox (“Records finished exporting to " + sFile)
End Sub
Function FieldName(lChannel As Long, sFormNo As String, sField As String)
As String
 Dim sQ As String
sQ = Chr(34)
 FieldName = DDERequest(lChannel, "[FormGetFieldName(" + sQ + sFormNo + sQ
+ "," + sQ + sField + sQ + ")]")
End Function
```

Deleting Fields from a Form

Syntax	[FORMCLEARFIELDS(<FormNo>)]

The FormClearFields function opens an existing form profile and deletes all associated fields.

Parameters

The FormClearFields function takes one parameter—the number of the form. Enclose this parameter in quotation marks (").

Return Value

The FormClearFields function returns *1* if the profile is open, or *0* if an error occurs.

Example

See Adding Merge Fields to a Form .

Closing a Form Profile

Syntax	[FORMCLOSEFORM(<FormNo>)]

The FormCloseForm function closes an open form profile.

Parameters

The FormCloseForm function takes one parameter, which is the number of the form. Enclose this parameter in quotation marks (").

Example

See Adding Merge Fields to a Form .

Creating an Xbase File with Registered Fields

Syntax	[FORMCREATEFILE(<FormNo>,<FileName>,<WhichRec>,<MergeCode>)]

The FormCreateFile function creates an Xbase (DBF) file with all registered fields. Any active filter or group that applies to the contact record is taken into account. FormCreateFile can be used to export data via DDE.

Parameters

The FormCreateFile function takes four parameters. Enclose all parameters in quotation marks (").

The first parameter is the number of the form.

The second parameter is the name of the .DBF file to be created.

The third parameter indicates which records are to be exported. The WhichRec value is the sum of values for each available listed below.

WhichRec Values

Value	Description
1	Primary
2	Secondary
4	All records
8	Forward to last
16	Return control to the calling program immediately after export has started

Examples of WhichRec Parameter

Current contact	1
All primary contacts	5 (1+4)
Forward to last of primary and additional contacts	11 (1+2+8)

The fourth parameter is the merge code. If any merge code value(s) are included in the function, only records with the matching merge code(s) will be included. To include multiple merge codes, place a space between each individual merge code. If the fourth parameter is empty, all records are included.

Return Value

The FORMCREATEFILE function returns the total number of records in the `output .DBF` file.

Example

See Adding Merge Fields to a Form .

Returning a Field Name for an Expression

Syntax	[FORMGETFIELDNAME(<FormNo>,<Field>)]

The FormGetFieldName function returns the field name for an expression, a macro, or a field.

Parameters

The FormGetFieldName function takes two parameters. Enclose both parameters in quotation marks (").

The first parameter is the number of the form. The second parameter is the name of the field, macro, or expression to be associated with the file name.

Example

See Adding Merge Fields to a Form .

Returning a Value for Unattached Fields

Syntax	[FORMNEWFORMNO()]

Return Value

The FormNewFormNo function returns a new, unique FormNo value that can be used to register fields not attached to a GoldMine form. Enclose this parameter in quotation marks (").

Example

See Adding Merge Fields to a Form

Counting the Number of Exported Records

Syntax	[FORMQUERYCREATE(<FLAGS>)]

The FormQueryCreate function provides status information during an export by returning the number of records exported during the export process.

Parameters

The FormQueryCreate function takes one optional parameter. Enclose this parameter in quotation marks (").

The following table lists values of FormQueryCreate parameters.

FormQueryCreate Parameters

Value	Description
0	Export in progress (default)
1	Start process
2	Abort process

Return Value

The FormQueryCreate function returns the number of records created while an export is in progress, or -1 when the record export process is completed.

Example

See Adding Merge Fields to a Form .

Creating a History Record

Syntax	[INSHISTORY(<accno>,<rectype>,<ref>,<notes>,<actv>,<rslt>,<user>)]

The InsHistory function is used to create a history record in GoldMine. The InsHistory function provides a higher level interface for creating these records than using Open, Append, and Replace.

Parameters

The InsHistory function takes up to seven parameters. All parameters must be passed in quotation marks (").

The first parameter is the account number of the contact record to which the new history record will be linked.

The second parameter is the record type to create. The following values are available:

InsHistory Valid Values (2nd parameter)

Value	Record Type	Value	Record Type
A	Appointment	U	Unknown
C	Phone call	CC	Call back
D	To-do	CI	Incoming call
E	Event	CM	Returned message

L	Form	CO	Outgoing call
M	Sent message	MG	E-mail message
O	Other	MI	Received e-mail
S	Sale	MO	Sent e-mai
T	Next action		

The third parameter is the history *Reference*.

The fourth parameter (optional) is the *Notes* for the history record.

The fifth parameter (optional) is the *Activity Code*.

The sixth parameter (optional) is the *Result Code*.

The seventh parameter is the *User*. If this parameter is not specified, the **User** field defaults to the currently logged user.

Return Value

The InsHistory function returns the record number (Xbase) or record ID (SQL) of the new history record if the function was completed successfully. The function returns 0 if a new record could not be appended to the data file.

Example

The following example shows how to create a history (incoming call) record for the current contact.

NOTE: The example below is written in Visual Basic for Applications, and the DDEInitiate and DDERequest functions are not a part of Visual Basic 6.0. DDE functionality is performed via the LinkRequest method in a textbox. For more information, see Establishing a DDE Conversation.

```
Sub Main()
Dim lChannel As Long
Dim sAccountNo As String
Dim sRecType As String
Dim sRef As String
Dim sRet As String
Dim sQ As String

sQ = Chr(34)
lChannel = DDEInitiate("GoldMine", "Data")
sAccountNo = DDERequest(lChannel, "Contact1->AccountNo")
sRecType = "CI" 'Incoming Call
sRef = "New History"
sRet = DDERequest(lChannel, "[InsHistory(" + sQ + sAccountNo + Chr$(34) +
"," + Chr$(34) + sRecType + Chr$(34) + "," + Chr$(34) + sRef + sQ + ")]")
If sRet = "0" Then
StatusBar = "History not Created"
End If
DDETerminate (lChannel)
EndSub
```

Creating or Updating a Document Link

Syntax	[LinkDoc(<recno>,<filepath>,<title>,<owner><notes>,<nSync>)]

The LinkDoc function is used to create or update a document link in GoldMine. Document links allow you to launch directly into an application and load the application with a document by clicking on the desired document listed in the contact's Links tab. GoldMine maintains these links as records in the supplementary data file. The LinkDoc function provides a higher level interface to these records than can be obtained by using Open, Append, and Replace.

Parameters

The LinkDoc function takes up to six parameters.

The first parameter is the record number of the link record to be updated. If a new link record is to be created, pass 0 as the first parameter.

NOTE: When GoldMine calls the mail merge macro, the record number of the linked document record is passed as the sixth parameter.

The second parameter is the fully qualified path and filename of the file to link. Keep in mind that a valid association must exist for the file's extension if GoldMine is to automatically launch the file's application. See "Installing the GoldMine DDE Link" for information on creating a file association using Windows Explorer. Enclose this parameter in quotation marks (").

The third parameter is the document title. Enclose this parameter in quotation marks (").

The fourth parameter is the optional document owner. If this field is not passed, the document owner defaults to the name of the currently logged GoldMine user.

The fifth parameter is optional notes for the linked document record in the *Links* tab.

The sixth parameter defines the remote synchronization status for the linked document from the values shown in the following table.

NSync Valid Values

Value	Action
-1	Uses the GoldMine default as defined *by Allow new documents to sync by default* in the *Sync* tab of the *Preferences* window.
0	Does not synchronize the newly linked document.
1	Allows the newly linked document to synchronize.

Return Value

The LinkDoc function returns the new record number (Xbase) or record ID (SQL) if the function was completed successfully. The function returns any empty string if a new record could not be appended to the data file, or an existing record could not be locked for update.

Example

The following example prompts the user for a file name and description, then creates a document link to the current contact.

Note that the example below is written in Visual Basic for Applications, and the DDEInitiate and DDERequest functions are not a part of Visual Basic 6.0. DDE functionality is performed via the LinkRequest method in a textbox. For more information, see Establishing a DDE Conversation.

```
Sub Main()
Dim lChannel As Long
Dim sDocPath As String
Dim sTitle As String
Dim sRet As String
Dim sQ As String

sQ = Chr$(34)
lChannel = DDEInitiate("GoldMine", "Data")
sDocPath = InputBox("Enter Full Path of Document to Link")
sTitle = InputBox("Enter Title of Link")
sRet = DDERequest(lChannel, "[LinkDoc( 0," + sQ + sDocPath + sQ + "," + sQ
+ sTitle + sQ + ")]")
DDETerminate (lChannel)
End Sub
```

Displaying a Message Dialog Box

Syntax	[MsgBox(<message>,<style>)]

The MsgBox function displays a standard Windows message dialog box.

Parameters

The MsgBox function accepts two parameters.

The first parameter is the message to display within the dialog box. Enclose this parameter in quotation marks (").

The second parameter is the optional style of the message box. This value is the sum of the following options:

MsgBox Style Values (2nd parameter)

Value	Meaning
0	Display OK button only
1	Display OK and Cancel buttons
2	Display Abort, Retry, and Ignore buttons
3	Display Yes, No, and Cancel buttons
4	Display Yes and No buttons

5	Display Retry and Cancel buttons
16	Display Stop icon
32	Display Question Mark icon
48	Display Exclamation Mark icon
64	Display Information icon
128	First button is default
256	Second button is default
512	Third button is default

Return Value

The MsgBox function returns the following values:

MsgBox Return Values

Return	Description
1	OK button selected
2	Cancel button selected
3	Abort button selected
4	Retry button selected
5	Ignore button selected
6	Yes button selected
7	No button selected

Example

The following example shows how to display a message dialog box in GoldMine and return the result.

NOTE: The example below is written in Visual Basic for Applications, and the DDEInitiate and DDERequest functions are not a part of Visual Basic 6.0. DDE functionality is performed via the LinkRequest method in a textbox. For more information, see Establishing a DDE Conversation.

```
Sub Main()
Dim lChannel As Long
Dim sRet As String
Dim sQ As String

sQ = Chr(34)
lChannel = DDEInitiate("GoldMine", "Data")
```

```
sRet = DDERequest(lChannel, "[MsgBox(" + sQ + "Press a Button, Any Button"
+ sQ + ", 4)]")
If ret$ = "6" Then
MsgBox ("Yes was pressed")
Else
MsgBox ("No was pressed")
End If
DDETerminate (lChannel)
End Sub
```

Adding a Merge Form

Syntax	[NEWFORM(<apptype>,<filepath>,<title>,<macro>, <templatetype>,<flags>)]

The NewForm function adds a merge template record into the Merge Forms window in GoldMine. This function is used primarily by the document merge link installation macro; however, the function can also be used to add additional merge templates from a user-written application.

Parameters

The NewForm function takes up to six parameters; the first three parameters are required, and the last three parameters are optional.

The first parameter is the type of document to which the new form record will point. This value must be a valid Application Identifier, such as Word.Document.6, that corresponds to an entry in the Registration Database. Enclose this parameter in quotation marks (").

The second parameter is the fully qualified path and filename of the template file.

The third parameter is the title of the document as it should appear in the Merge Forms browse window. Enclose this parameter in quotation marks (").

The fourth parameter is the name of an optional DDE function to be called after the template is loaded by the linked application. If this parameter is not specified, the default function is MAINMENU. Enclose this parameter in quotation
marks (").

The fifth parameter is the optional type of template. If this parameter is not specified, the template type is assumed to be Document. Enclose this parameter in quotation marks (''). GoldMine accepts the following values for this parameter:

Document Types

Type	Description
0	Document
1	Spreadsheet
2	Other

The sixth parameter is a three-character field corresponding to the values of the *Link To Doc*, *Save History* and *Allow Hot Link* options on the *Form Setup* dialog box. To set (check) one of these options, *1* is passed; to reset (uncheck), *0* is passed. Enclose this parameter in quotation marks (“).

Flag Values

Position	Description
0	Link To Doc check box
1	Save History check box
2	Allow Hot Link check box

Return Value

The NewForm function returns a form number.

Example

The following example shows how to create a merge form entry in GoldMine, using the currently active Word Document.

NOTE: The example below is written in Visual Basic for Applications, and the DDEInitiate and DDERequest functions are not a part of Visual Basic 6.0. DDE functionality is performed via the LinkRequest method in a textbox. For more information, see Establishing a DDE Conversation.

```
Public Sub Main()
Dim sQ As String
Dim lChannel As Long
Dim iResult As Integer
Dim sDocTitle As String
Dim sFullName As String
Dim sAppName As String
Dim FSDlg As Dialog

'GoldMine Is Not running.
SQ = Chr(34)
If Not (Tasks.Exists("GoldMine")) Then
MsgBox Prompt:="GoldMine is NOT Running", Buttons:=vbCritical, Title:="Save
As Merge Form"
GoTo Bye
End If
lChannel = DDEInitiate("GoldMine", "Data")
iResult = Dialogs(wdDialogFileSummaryInfo).Show
If iResult = 0 Then
GoTo Bye
End If
sDocTitle = sQ + Dialogs(wdDialogFileSummaryInfo).Title + sQ
iResult = Dialogs(wdDialogFileSaveAs).Show
If iResult = 0 Then
GoTo Bye
End If
```

```
ActiveDocument.Save
sFullName$ = sQ + ActiveDocument.FullName + sQ
sAppName = sQ + "[GoldMineLink()]" + sQ
FormNo$ = DDERequest(lChannel, "[NewForm(Word.Document.8," + sFullName$ +
"," + sDocTitle$ + "," + sAppName + ")]")
ActiveDocument.Saved = False
ActiveDocument.SaveAs FileName:=sFullName$, FileFormat:=wdFormatTemplate
StatusBar = "Document has been saved as a GoldMine Merge Form"
Bye:
If lChannel Then
DDETerminate lChannel
End If
End Sub
```

Creating a Group

Syntax	[NEWGROUP(<ref>,<code>,<user>)]

The NewGroup function is used to create an empty group. This function must be called prior to adding group members with the NewMember function.

Parameters

The NewGroup parameter takes up to three parameters; the first parameter is required, the last two are optional.

The first parameter is the Reference for the new group. Enclose this parameter in quotation marks (“).

The second parameter is the optional sort Code for the group. This parameter must be passed in quotation marks if it contains any embedded spaces or delimiting marks.

The third parameter is the optional user name to whose groups list the new group will be added. If this parameter is not passed, the new group will be added to the currently logged user’s list of groups. Enclose this parameter in quotation marks (“).

Return Value

The NEWGROUP function returns a value representing the GROUP NUMBER of the new group. Zero is returned if the group could not be added. The GROUP NUMBER value is used by the NewMember function to add members to the new group.

Example

The following example shows how to create a group called “New Group” and make the current contact a member of that group.

NOTE: The example below is written in Visual Basic for Applications, and the DDEInitiate and DDERequest functions are not a part of Visual Basic 6.0. DDE functionality is performed via the LinkRequest method in a textbox. For more information, see Establishing a DDE Conversation.

```
Sub Main()
Dim lChannel As Long
Dim sGroupNo As String
Dim sAcountNo As String
```

```
Dim sQ As String
Dim sRet As String

sQ = Chr(34)
lChannel = DDEInitiate("GoldMine", "Data")
sGroupNo = DDERequest(lChannel, "[NewGroup(" + sQ + "New Group" + sQ + "," 
+ sQ + "New" + sQ + ")]")
If sGroupNo <> “0” Then
sAccountNo = DDERequest(lChannel, "Contact1->AccountNo")
sRet = DDERequest(lChannel, "[NewMember(" + sQ + sGroupNo + sQ + "," + sQ +
sAccountNo + sQ + "," + sQ + "New Member" + sQ + "," + sQ + "Sort" + sQ +
")]")
If sRet = "" Then
StatusBar = "Error Creating New Member"
Else
StatusBar = "Group Created and Member Added. "
End If
Else
StatusBar = "Error Creating New Group"
End If
DDETerminate (lChannel)
End Sub
```

Adding a Group Member

Syntax	[NEWMEMBER(<groupno>,<accno>,<ref>,<code>)]

The NewMember function is used to add a member to a group created with the NewGroup function.

Parameters

The NewMember function takes up to four parameters; the first two parameters are required, and the last two are optional.

The first parameter is the GROUP NUMBER of the group to which the member will be added. This value is returned by the NewGroup function. Enclose this parameter in quotation marks (").

The second parameter is the account number of the contact record to add to the group. Enclose this parameter in quotation marks (").

The third parameter is the optional group member Reference. Enclose this parameter in quotation marks (").

The fourth parameter is the optional group member sort Code. Group members are ordered alphabetically by the value in this field. Enclose this parameter in quotation marks (").

Example

See Creating a Group .

Creating a Macro

Syntax	[PLAYMACRO(<Macro>,<wait>)]

A macro groups together a series of commands, keystrokes, and/or mouse clicks into a one-step operation. You can create a macro to automate a sequence of tasks that you perform frequently in GoldMine.

Parameters

The PlayMacro function takes two parameters that identify the macro and assign a wait state.

The first parameter identifies the macro. Either the number for the currently logged user or a valid macro filename can be used to identify a macro.

Identifying a Macro by Number

Each user can create up to 100 macros from the GoldMine toolbar. Each macro can be assigned an optional numeric identification from 800 to 899. For example, you can assign 800 to identify your first macro, 801 to identify your second macro, and so on.

> **TIP:** For details about creating a macro from the GoldMine toolbar, see "Customizing Toolbars" in the online Help.

Identifying a Macro by File Name

You can assign a file name to identify the macro, such as C:\GOLDMINE\MACROS\JOHN.801.

The second parameter assigns a wait state that determines GoldMine availability to process another macro or task while the current macro executes. To set GoldMine to wait for the currently executing macro to finish before starting another task, set the parameter to 1. For example, if you are setting up a sequence of macros to run tutorial lessons, you want GoldMine to wait for each lesson to finish before executing the next macro that will run the following lesson.

To allow GoldMine to perform background processing, such as indexing, while the macro(s) execute, set the parameter to 0.

Return Value

The PlayMacro function returns an integer value based on the wait parameter; that is, GoldMine availability to process a task in addition to the currently running macro. If the wait parameter is 0 (GoldMine does not wait for the macro to finish to process another task), the PlayMacro function will always return 1. If the wait parameter is 1 (GoldMine will wait for the current macro to finish before processing another macro or task), the PlayMacro function will return either 0 or 1 under the following conditions:

PlayMacro Return Values

Return	Description
0	Error occurred during macro playback
1	Macro played successfully

Example

The following example shows how to play back a macro via DDE.

> **TIP:** To prevent unwanted macros from being executed, some parts of this example have been commented out.

NOTE: The example below is written in Visual Basic for Applications, and the DDEInitiate and DDERequest functions are not a part of Visual Basic 6.0. DDE functionality is performed via the LinkRequest method in a textbox. For more information, see Establishing a DDE Conversation.

```
Sub Main()
Dim lChannel As Long
Dim sRet As String
Dim sQ As String

sQ = Chr(34)
'un comment the following line to execute
'lChannel = DDEInitiate("GoldMine", "Data")

'Play macro 800 for current user
sRet = DDERequest(lChannel, "[PlayMacro(800,0)]")

'Play Macro 802 for specified use (BILL)
sRet = DDERequest(lChannel, "[EXPR(" + sQ + "C:\GOLDMINE\MACROS\BILL.802" +
sQ + ")]")

End Sub
```

To Play a Macro from the Command Line

You can also play a macro from the command line (DOS prompt). Executing a macro from the command line can be useful in running functions at night, such as indexing, running an Automated Process, or synchronizing with remote sites with a transfer set created via macro. You can either identify a macro by an identification number, like GMW4 /m:801, or by file name like GMW4 /m:c: \index.801. If necessary, the command line statement can start GoldMine and then, once started, run the macro.

Optional switches include:

/m: Logs in automatically to GoldMine

/u:[username] Provides the username entry to log in to GoldMine

/p:[password] Provides the password entry to log in to GoldMine

If running the Plus! Pack with Windows, you can run a macro from the System Agent by placing a command line switch for GoldMine in the Program field of the Schedule a New Program dialog box that will run a macro. For example, to log in John with his username and password, then run John's first macro, place the following macro in the System Agent:

GMW5 /u:john /p:pswd /m:800

Where *GMW5/* starts Goldmine, *u:john/* is login user John, *p:pswd/* enters the password password, and *m:800* runs first macro.

Creating and Sending a Pager Message

Syntax	[SENDPAGE(<Message>,<From>,<To>)]

The SendPage function allows you to create and send a message to the pager of a GoldMine user. The function consists of the following components:

<Message> can consist of any text message that you create with this function to send to a pager; most pages can accept messages of 70–100 characters.

<From> includes the sender's name as an optional "signature."

<To> identifies an optional GoldMine user who will receive the pager message. Information about the pager must be entered in the Edit|Preferences|Pager tab, such as ID code or PIN number, telephone number of the pager, and maximum message size in characters that the pager can accept.

Return Value

The SendPage function can return one of two values.

SendPage Return Values

Return	Description
0	Error occurred during the attempt to send the message to the pager
1	Pager message was transmitted successfully

Example

The following example sends the message "This is a pager message" from John Doe:

NOTE: The example below is written in Visual Basic for Applications, and the DDEInitiate and DDERequest functions are not a part of Visual Basic 6.0. DDE functionality is performed via the LinkRequest method in a textbox. For more information, see Establishing a DDE Conversation.

```
Sub Main()
Dim lChannel As Long
Dim sMsg As String
Dim sFrom As String
Dim sRet As String
Dim sQ As String

sQ = Chr(34)
lChannel = DDEInitiate("GoldMine", "Data")
sMsg = "This is a pager message"
sFrom = "Jon Doe"
sRet = DDERequest(lChannel, "[SendPage(" + sQ + sMsg + sQ + "," + sQ +
sFrom + sQ + ")]")
End Sub
```

Displaying a Message in the GoldMine Status Bar

Syntax	[StatusMsg(<message>,<delay>)]

The StatusMsg function displays a message in the GoldMine status bar.

Parameters

The StatusMsg function takes two parameters. Enclose each parameter in quotation marks (").

First parameter is the message.

Second parameter is an optional delay, after which time the message is removed from the status bar.

Example

See “RecNo” on page 105.

Converting TLog Timestamps

Syntax	[SyncStamp(<stamp>)]

The SyncStamp function converts a TLog timestamp to a date and time representation, and from a date and time representation back to the TLog time stamp format.

Parameter

The SyncStamp function takes one parameter. Enclose the parameter in quotation marks (").

Return Values

When the <stamp> string parameter is exactly 17 characters long, formatted as Date:Time in form of CCYYMMDD:HH:MM:SS, the return string is in TLog time stamp format, exactly seven characters long. When the <stamp> parameter is seven characters long, and formatted as a TLog timestamp, the return string is formatted as CCYYMMDD:HH:MM:SS. An empty return string indicates an error.

Example 1

The following example converts February 1, 1998 at 7:01 p.m. to a TLog time stamp format.

[SyncStamp("19980201:19:01:30")] returns "+#G><N2"

Example 2

The following example converts a TLog time stamp format to the date and time of February 1, 1998 at 7:01 p.m.

```
[SyncStamp("+#G><N2")]
returns "19980201:19:01:30"
```

DDE Macros

To facilitate the use of DDEAUTO fields, GoldMine allows you to select a macro as the service item. Upon encountering a DDE service item that starts with an ampersand (&), GoldMine searches an internal table of macro names. If a match is found, the macro is processed and the result is returned, as if a DDE function or expression had been used.

Most macros are sensitive to the setting of the RECORDOBJ function’s SETRECORD subfunction. This DDE function is used primarily to gain access to additional contacts and other supplementary information. When the SETRECORD type is set to PRIMARY, the following macros will return the value from the corresponding fields in the primary information portion of the contact record. When the SETRECORD type is set to CONTACTS (additional contacts), or another supplementary record type, the macros will return the value from the corresponding field in the supplementary file (CONTSUPP.DBF).

The following macros can be used as DDE service items:

&Address	Returns a string containing the values of both *&Address1* and *&Address2*, separated by a carriage return and line feed character. If either *&Address1* or *&Address2* does not contain any data, a single line of data is returned, without the carriage return and line feed character. This macro can be used to perform rudimentary blank line suppression within linked applications that do not support blank address line suppression internally. The action of this macro string is similar to the action of the *&Address* macro. The *&Address2* macro can be used to return an additional contact address by using the RECORDOBJ SETRECORD subfunction.
&Address1	Returns the first *Address* field from the active contact record. Typically, this value will be extracted from the *Address1* field in the primary display portion of the contact record; however, when the RECORDOBJ SETRECORD subfunction has been used to change the returned record type to CONTACTS, then GoldMine returns the value from the *Address1* field on the additional contact record, if a value is entered. When the **Address1** field on the additional contact record is blank, then the *&Address1* macro returns the value in the *Address1* field in the primary display portion of the contact record. When the RECORDOBJ SETRECORD type is set to return a record type other than CONTACTS, the *&Address1* macro returns the value in *Address1* field in the primary display portion of the contact record.
&Address2	Returns the second *Address* field from the active contact record. Typically, this value will be extracted from the *Address2* field in the primary display portion of the contact record; however, when the RECORDOBJ SETRECORD subfunction has been used to change the returned record type to ADDITIONAL, then GoldMine returns the value from the *Address2* field on the additional contact record, if an entry exists in the **Address2** field on the additional contact record. When the *Address2* field on the additional contact record is blank, then the *&Address2* macro returns the value in the **Address2** field in the primary display portion of the contact record. When the RECORDOBJ SETRECORD type is set to return a record type other than PRIMARY or ADDITIONAL, the *&Address2* macro returns the value in the *Address2* field of the primary display portion of the contact record.
&BrowseRecNo	*Xbase*: Returns the record number of the last selected record in a browse window. *SQL*: Returns the record ID of the last selected record in a browse window.
&CalRefresh	Refreshes the graphical calendar display. Set up GoldMine to run this macro after adding calendar records using DDE.
&City	Returns the *City* field from the active contact record. The action of this macro string is similar to the action of *&Address1*. The *&City* macro can be used to return an additional contact city by using the RECORDOBJ SETRECORD subfunction.

&CityStateZip	Returns a format string of text containing the *City*, *State*, and *Zip* fields from the active contact record. This string is returned in the following format: City, State Zip The action of this macro string is similar to the action of *&Address1*. The *&CityStateZip* macro can be used to return an additional contact city, state, and ZIP Code by using the RECORDOBJ SETRECORD subfunction.
&CommonDir	*Xbase*: Returns the path information for the directory where the contact sets are located. *SQL*: Returns the BDE alias where the contact sets are located.
&Contact	Returns a Contact name from the active contact record. Normally, this value will be extracted from the Contact field in the primary display portion of the contact record; however, the RECORDOBJ SETRECORD subfunction can be used to change the returned record type to additional contact, or another type of supplementary record. When the RECORDOBJ SETRECORD type is set to return record types other than PRIMARY, the &Contact macro returns the value in Contact field in CONTSUPP for the current supplementary record.
&Country	Returns the Country field from the active contact record. The action of this macro string is similar to the action of &Address1. The &Country macro can be used to return an additional contact country by using the RECORDOBJ SETRECORD subfunction.
&Dial1	Returns the Phone1 entry from the active contact record. The returned phone number is formatted for dialing. GoldMine applies the same rules used to dial the phone via TAPI. If selected, PREDIAL.INI settings are applied to phone number selection.
&Dial2	Returns the Phone2 entry from the active contact record. For details, see &Dial1 above.
&Dial3	Returns the Phone3 entry from the active contact record. For details, see &Dial1 above.
&DialFax	Returns the FAX entry from the active contact record. For details, see &Dial1 above.
&EmailAddress	Returns the primary e-mail address for the currently selected contact.
&Fax	Returns the fax number as it should be sent to an auto-dialer for automatic fax transmission.
&Filter	Returns the activated filter expression.
&FirstName	Returns the first name of the current contact.

&FullAddress	Returns a string containing the complete address for the contact record, composed of values of &Address1, &Address2, &City, &State, and &ZIP. The action of this macro string is similar to the action of &Address1. The &FullAddress macro can be used to return an additional contact address by using the RECORDOBJ SETRECORD subfunction.

&GetRoTabID	Returns the ID of the currently selected tab. Typically, this value will verify that the correct tab is selected when a user starts a custom application.
	
	The following values are valid:
	
	0 = Summary
	1 = Fields
	2 = GM+View
	3 = Notes
	4 = Contacts
	5 = Details
	6 = Referral
	7 = Pending
	8 = History
	9 = Links
	10 = Members
	11 = APs/Tracks
	12 = Opportunities
	13 = Projects
	14 = Relationships/Org tree
	15 = Cases
	16 = HEAT View if installed, else it will go to the first tab
	17+ = custom if installed, otherwise the first tab
	
	The following example tests the selection of the **Details** tab:
	ch=DDEInitiate(“GoldMine”, “Data”)
	If DDERequest$(Ch, “&GetRoTabID”) <> “6” Then
	MsgBox “You must select a detail record first”
	End If

&GetRoTabPos	Returns the currently selected tab position. Since the tabs can be rearranged, this method is not always reliable for determining the currently selected tab. For details, see &GetRoTabID.
&GoldDir	*Xbase*: Returns path information for the directory in which GoldMine is installed. *SQL*: Returns path information for BDE alias in which GoldMine is installed.
&LastFirstName	Returns the name of the current contact in the format: last name, first name
&LicUsers	Returns the number of concurrent users allowed to log in to the installed copy of GoldMine.

&LicUsersAvailable	Returns the number of users allowed to log in to the installed copy of GoldMine license.
&NameAddress	Returns a string containing the contact's name, company, and complete address of the current contact record. Each address line is separated by a carriage return and line feed, and the entire string is formatted so that the string can be inserted directly into a merge template. If any of the address lines on the contact record is empty, that address line will be suppressed. This macro can be used to perform rudimentary blank line suppression within linked applications that do not support blank address line suppression internally. The action of this macro string is similar to the action of the &ADDRESS macros, and the &NAMEADDRESS macro can be used to return an additional contact address by using the RECORDOBJ SETRECORD subfunction.
&NameTitleAddress	Returns a string containing the contact's name, title, department, company, and complete address of the current contact record. Each line is separated by a carriage return and line feed, and the entire string is formatted so that the string can be inserted directly into a merge template. If any of the lines on the contact record is empty, that line will be suppressed. This macro can be used to perform rudimentary blank line suppression within linked applications that do not support blank address line suppression internally. The action of this macro string is similar to the action of the &ADDRESS macros, and the &NAMETITLEADDRESS macro can be used to return an additional contact address by using the RECORDOBJ SETRECORD subfunction.
&NewRecID	Returns a unique record ID, which can be used when creating new records.
&Notes	Returns the *Notes* from the active contact record. Typically, this value will be extracted from the *Notes* field in the primary display portion of the contact record; however, the RECORDOBJ SETRECORD subfunction can be used to change the returned record type to additional contact, or another type of supplementary record. When the RECORDOBJ SETRECORD type is set to other than PRIMARY, the &TITLE macro returns the value in Notes field in CONTSUPP for the current supplementary record.
&Phone	Returns a telephone number from the selected contact record. The action of this macro string is similar to the action of the &ADDRESS1. The &PHONE macro can be used to return an additional contact telephone number by using the RECORDOBJ SETRECORD subfunction.

&Profile(s)	Two related macros:
	&Profile: Returns the first matching profile record for the selected contact.
	&Profiles: Returns all profile records for the selected contact.
	Both of these macros take optional parameters. Each parameter must be separated by a period (.). Although GoldMine does not typically pass parameters with a DDE macro, the structure of &Profiles must be different for DDE fields in Microsoft Word document templates, which do not take DDE commands.
	The following examples show the syntax for the &Profile(s) macros:
	`&Profile Example 1`
	`&Profile.ProfileName.Reference.Flags`
	Retrieves the first profile that matches the ProfileName and Reference.
	In both of the above examples, the Reference parameter is optional. If passed, the Reference parameter acts as a "begin with" condition on the profile reference. If the Reference parameter is not passed, all ProfileName profiles are evaluated.
	The optional Flags parameter has the following values:
	2 Returns the extended profile fields
	4 Returns the ProfileName and Reference
	The &Profile(s) macro can easily fill in a Word table with the selected contact's profile information because tabs separate each field value, and a CR/LF separates each profile record.
	`&Profile Example 2`
	The following example returns the first e-mail address of the contact:
	`&Profile.E-mail Address`
	`&Profiles Example 1`
	The following example returns all the computer profiles that begin with the word notebook:
	`&Profiles.Computer.Notebook`
	`&Profiles Example 2`
	The following examples use the Flags parameter to specify the profile fields to return:
	`&Profiles.Computer.Notebook`
	`Notebook ThinkPad 770|`
	`Notebook Compaq Elite|`
	`Notebook Dell 1200|`
	
	`&Profiles.Computer.Notebook.2`
	`Computer|Notebook ThinkPad 770|`
	`Computer|Notebook Compaq Elite|`
	`Computer|Notebook Dell 1200||`
	
	`&Profiles.Computer.Notebook.4`
	`Computer|Notebook ThinkPad 770|IBM|233Mz|`
	`Computer|Notebook Compaq Elite|Compaq|200mz|`
	`Computer|Notebook Dell 1200|Dell|166mz|`

&RoTabPage	Returns the currently selected tab. Typically, this value will verify that the correct tab is selected when a user starts a custom application. Values between 1 and 9 represent tabs in the first row of tabs; for example, 1 represents the *Summary* tab . Values between 10 and 18 represent tabs in the second row, and 19–27 represent tabs in the third row.
	The following example tests the selection of the fifth (*Profiles*) tab:
	`ch=DDEInitiate("GoldMine", "Data")`
	`If DDERequest$(Ch, "&RoTabPage") <> "5" Then`
	`MsgBox "You must select a profile record first"`
	`End If`

&SerialNo	Returns the serial number of the installed GoldMine program.

&SetRoTab#	Selects the tab that corresponds to the number (represented by #) in the active contact record.
	The following values are valid:
	1 = Summary
	2 = Fields
	3 = GM+View
	4 = Notes
	5 = Contacts
	6 = Details
	7 = Referral
	8 = Pending
	9 = History
	10 = Links
	11 = Members
	12 = APs/Tracks
	13 = Opportunities
	14 = Projects
	15 = Relationships/Org tree
	16 = Cases
	17 = HEAT View if installed, else it will go to the first tab
	18+ = custom if installed, otherwise the first tab
	
	`ch=DDEInitiate("GoldMine", "Data")`
	`DDERequest$(Ch, "&SetRoTab4")`
	
	Displays the *Notes* tab in the contact record.

&ShutDown	Logs out the currently logged user, and quits GoldMine.
&State	Returns the *State* field from the active contact record. The action of this macro string is similar to the action of the &ADDRESS1. The &STATE macro can be used to return an additional contact state by using the RECORDOBJ SETRECORD subfunction.
&SysDir	Returns the GoldMine system directory.
&SysInfo	Displays system information as returned by Help>About GoldMine>System Info.

&Title	Returns the *Title* from the active contact record. Normally, this value will be extracted from the *Title* field in the primary display portion of the contact record; however, the RECORDOBJ SETRECORD subfunction can be used to change the returned record type to additional contact, or another type of supplementary record. When the RECORDOBJ SETRECORD type is set to other than PRIMARY, the &TITLE macro returns the value in Title field in CONTSUPP for the current supplementary record.

&User_Var	Returns the defined field value from all users, a specified user, or the currently logged user. For details on defining values, see "Defining Field Values for use with External Applications" in Maintaining GoldMine.
	The &User_Var macro allows GoldMine users to store specific data that can be retrieved later into applications that are linked via DDE with GoldMine. This macro can be defined in the [user_var] section of both the GM.INI and the username.INI of GoldMine.
	Usage Syntax:
	`&User_Var.<variable name>.<GoldMine username>`
	Example:
	`&User_Var.Territory.Dan`
	(Where *<variable name>* is a descriptive name of the macro and *<GoldMine username>* assigns a defined value to a specific GoldMine user.)
	<GoldMine username> is optional, as GoldMine will assign these values to the current GoldMine user.

&UserFullName	Returns the full name of the currently logged GoldMine user as the name appears in the *FullName* field in the *Users Master File* for the user.
&UserName	Returns the login name of the currently logged GoldMine user.
&Version	Returns the version number of the installed GoldMine program.
&WebSite	Returns *http://<Web site>* for the active contact.
&ZIP	Returns the Zip field from the currently active contact record. The action of this macro string is similar to the action of the &ADDRESS1. The &ZIP macro can be used to return an additional contact ZIP Code by using the RECORDOBJ SETRECORD subfunction. The DDE macro can be used to reindex or rebuild the database.

DDE Macros for Merge Forms

The following DDE macros are used primarily for creating DDE links to GoldMine through the Merge Forms function. The values returned by each of these macros are updated by GoldMine when a Merge Form is launched by selecting *Edit*, *Link*, *Print* or *Fax* from the *Merge Forms* dialog box.

&PARAM1 (filename)	Returns the path and filename of the document template associated with the merge form selected when **Edit, Link, Print,** or **Fax** was selected. This value is obtained from the **Template File** field in the merge form's **Form Setting** dialog box.

&PARAM2 (action)	Returns a value indicating whether the **Edit, Link, Print,** or **Fax** button was selected to launch linked application.

&PARAM2 Parameters

Value	Description
1	*Edit* selected
2	*Link* selected
3	*Print* selected
4	*Fax* selected

&PARAM3 (range)	Returns a value corresponding to the setting of the *Record Range* options on the *Merge Forms* dialog box when the *Edit*, Link, *Print*, or *Fax* button was selected.

&PARAM3 Parameters

Value	Description
1	**This contact** selected
2	**All contacts** selected
3	**Forward to last** selected

&PARAM4 (scope)	Returns a value corresponding to the setting of the *Primary* and *Additional* check boxes on the *Merge Forms* dialog box when the *Edit*, *Link*, *Print*, or *Fax* button was selected.

&PARAM4 Parameters

Value	Description
1	**Primary** checked
2	**Additional** checked
3	Both **Primary** and **Additional** checked

&PARAM5 (flags)	Returns a value corresponding to the status of the *Link to Doc*, *Save History*, and/or *Allow Hot Link* check boxes on the *Merge Forms* dialog box. In addition, the returned value determines whether the form was merged as the result of an Automated Processes action. Returns a seven-character string. Each position of the string can contain either *0*, indicating the item was not checked (or Automated Processes is not active), or *1*, indicating the item was checked (or Automated Processes is active).

&PARAM5 Parameters

Position	Description
1	Link to Doc
2	Save History
3	Allow Hot Link
4	Unused
5	Unused
6	Unused
7	Automated Processes status

&PARAM6 (LinkDoc record number)	Returns a value containing the record number of the last Linked Document supplementary record created as a result of launching a Merge Form. When you launch a merge form with *Link to Doc* selected, GoldMine creates a linked document record to hold the saved document. This value can be saved and used to update the linked document record by passing the record number to the LinkDoc DDE function.
&PARAM7 (contact record pointer)	Returns a pointer to a minimized contact record that is created when *Print* or *Fax* is selected on the *Merge Forms* dialog box, and the **Record Range** is *All Contacts* or *Forward to Last*. This value can then be passed to the RecordObj function to further control a document merge from the linked application.
&PARAM8 (merge code value)	Returns the merge code entered in the *Merge code* field of the *Merge Forms* dialog box.
&PARAM9 (history record)	Returns the RecNo or RecID of the history record created by GoldMine. This macro is useful for updating the history record.

DDE Macros for the GoldMine License

The following DDE macros return data for the current GoldMine license. The descriptions for each macro include the corresponding field name from the form that appears in the Registration tab of the GoldMine Net-Update window. For details on the Net-Update process, see “Using Net-Update” in the online Help.

&LicInfoLicTo	Returns the *Organization* entry from the registration form.
&LicInfo_Contact	Returns the *Contact Name* entry from the registration form.
&LicInfo_LicEmail	Returns the *E-mail address* entry from the registration form.
&LicInfo_Phone	Returns the telephone number entry from the first *Phone/Fax* field.
&LicInfo_Fax	Returns the fax number entry from the second *Phone/Fax* field.
&LicInfo_Address1	Returns the *Address1* entry from the registration form.
&LicInfo_Address2	Returns the *Address2* entry from the registration form.
&LicInfo_City	Returns the city entry from the first *City/State* field.
&LicInfo_State	Returns the state or province entry from the second *City/State* field.
&LicInfo_Zip	Returns the ZIP Code entry from the first *Zip/Country* field.
&LicInfo_Country	Returns the country entry from the second *Zip/Country* field.

Using GMXS32.DLL for Database Access and Sync Log Updates

Overview

The GoldMine `GMXS32.DLL` is a standard dynamic-link library (DLL) that offers developers efficient methods to access GoldMine databases and update GoldMine synchronization logs when external applications update GoldMine data. Most development environments can load `GMXS32.DLL`. GoldMine does not need to run to use GMXS32.DLL.

GMXS32.DLL installs into the `\WINDOWS\SYSTEM` directory automatically with GoldMine. Therefore, third-party developers do not need to distribute `GMXS32.DLL` with their applications.

The actual file name for the API will vary depending on the version of GoldMine. For versions of GoldMine in the 5.0 ranges, the dll is named `GM5S32.DLL`. For versions in the 6.0 ranges, the dll is named `GM6S32.DLL`

For an in-depth discussion on interfacing with GoldMine, visit the *Public.GoldMine.Programming* newsgroup, which you can access directly from the GoldMine Web site at http://www.goldmine.com.

This document contains the information you need to:

- Load and initialize `GMXS32.DLL`
- Streamline integration with GoldMine
- Work with DataStream functions
- Work with low-level data access functions
- Update GoldMine synchronization information when data is changed by an external application not utilizing the GoldMine API.

Passing Multiple Parameters to a Function

Each Name/Value (NV) set, or container, simply combines a “Name” and a “Value.” In the following example:

Company=GoldMine, Inc

Company is the *Name* and GoldMine, Inc is the *Value*.

Using a set of NV pairs provides an easy mechanism to pass multiple parameters to a function. The user can populate the NV pairs into a container, then execute a Business Logic transaction against the container. The transaction adds extra pairs to the container to return the results.

Since the NV container remains in memory until cleared, it can make several calls without clearing all the previous values. This capability is useful to call the same function with only slight changes to the values, such as when a return value of one call is needed for a subsequent call.

Using the Business Logic methods, a developer can easily read and write GoldMine data. Previously, integrating with GoldMine required a great familiarity with the schema and methodology of GoldMine databases. The Business Logic functions require less direct knowledge and provide a more standardized and secure way to integrate with GoldMine. Business Logic functions wrap several other low-level calls to perform common tasks. In addition, the Business Logic functions take user security restrictions into account when reading and updating GoldMine data.

Comparing Low Level/DDE Methodology to Business Logic Methodology

We can compare an example flow to a common task using low level/DDE or Business Logic. In the following example, you can see that Method 2 has a simpler flow than Method 1.

Method 1: Updating a Contact Record using the low level functions or DDE

1. Open the Contact1 database.
2. Set the index tag.
3. Seek the contact record.
4. If not found, then Append a new record.
5. Replace field values.
6. Close the database.

Method 2: Updating a Contact Record using the Business Logic

1. Load an NV Container with the values for the contact record.
2. Execute the WriteContact method.

Loading GMXS32.DLL and Logging In

The following section describes the functions available to load the BDE and log in to a GoldMine table. For function prototypes and code examples in C++, Visual Basic and Delphi, see the .

If using C/C++, note that the `GMXS32.DLL` functions use the stdcall convention.

Before using any of the functions, you must perform the following steps:

1. `GMXS32.DLL` must be dynamically loaded in C/C++ (simply declare them in VB).
2. GMW_LoadAPI function must be called to load the API with the set parameters for the programmer to work with.

The GMW_UnloadAPI() function must always be called before terminating the application and freeing the DLL.

The following functions initialize and close the API sessions:

- *GMW_LoadAPI*: loads set parameters for an API session

- *GMW_UnloadAPI*: closes the API session

NOTE: As of GoldMine Version 7.0, the Borland Database Engine is no longer used. References to BDE in the following sections apply only to integrations developed in GoldMine Version 6.7 or lower.

For GoldMine Version 6.7 or Lower

The GMW_LoadBDE function must be called to load the BDE and initialize the database objects. The GMW_UnloadBDE() function must always be called before terminating the application and freeing the DLL.

The following functions initialize and close the BDE sessions:

- **GMW_LoadBDE**: loads a BDE session
- **GMW_UnloadBDE**: closes the BDE session

Setting the SQL Database Login Name and PasswordGoldMine 6.7 or lower only)

This topic pertains to SQL only. GMW_SetSQLUserPass should be called immediately prior to the GMW_LoadBDE call. GMW_SetSQLUserPass is required only when accessing SQL tables, and will have no effect on Xbase tables. This function is not required if using DDE login credentials with versions of GoldMine beyond 5.70.20222.

Syntax

C/C++	int _stdcall GMW_SetSQLUserPass(char *szUserName, char *szPassword)
VB	Public Declare Function GMW_SetSQLUserPass Lib "gm6s32.dll" (ByVal strUserName As String, ByVal strPassword As String) As Long

Parameters

The GMW_SetSQLUserPass function takes two parameters:

- *szUserName*: specifies the SQL login name.
- *szPassword*: specifies the SQL login name's password.

Return Values

The GMW_SetSQLUserPass function returns the following values:

GMW_SetSQLUserPass Return Values

Return	Description
0	Failure
1	Success

Example

```
GMW_SetSQLUserPass("JON", "MyPASSWORD");
```

Loading an API Session (GoldMine 7.0 or higher)

Syntax

C/C++	int GMW_LoadAPI(char *szSysDir, char *szGoldDir, char *szCommonDir, char *szUser, char *szPassword)
VB	Public Declare Function GMW_LoadAPI Lib "gm6s32.dll" (ByVal strSysDir As String, ByVal strGoldDir As String, ByVal strCommonDir As String, ByVal strUser As String, ByVal strPassword As String) As Long

Parameters

The GMW_LoadAPI function takes five parameters.

SzGoldDir: Specifies the location of CAL.DBF or the database alias name to use as the main database.

NOTE: The database alias name must be appended with a colon (":").

SzCommonDir: Specifies the location of CONTACT1.DBF or the database alias name to use as the contact set database.

NOTE: The database alias name must be appended with a colon (":").

SzUser: Specifies the GoldMine user name (must be UPPERCASE).

For API version 5.70.20222 and later: You may set this parameter to the value of *DDE_LOGIN_CREDENTIALS* to use login credentials returned for the user logged into a running copy of GoldMine through DDE. For GoldMine 6.7 or higher, you may also use the UI API equivalent.

SzPassword: Specifies the user's password (must be UPPERCASE).
For API version 5.70.20222 and later: You may set this to the return string from the GetLoginCredentials DDE command if the User parameter is set to *DDE_Login_Credentials*. The credential string is only valid for 30 seconds.

Return Values

The GMW_LoadAPI function returns the following values:

GMW_LoadBDE Return Values

Return	Description
1	Success
0	API already loaded
-1	API failed to load
-2	Cannot find license file
-3	Cannot load license file
-4	Cannot validate the license file username/password

-5	Invalid GoldDir
-6	Invalid CommonDir
-7	Failed to allocate the needed TLS slot
-8	General Failure
-9	No access to specified contact set for this user

Notes

GMW_LoadAPI must be called before calling any function that accesses databases, such as GMW_UpdateSyncLog and GMW_ReadImpTLog. GMW_UnloadAPI must be called before unloading the DLL. GMW_LoadAPI may be called as many times as necessary. Be sure to match a corresponding GMW_UnloadAPI for every call of GMW_LoadAPI.

Example

```
GMW_LoadAPI( "d:\\GM4", "d:\\GM4", "d:\\GM4\\demo", "JON", “PASS” );
Or
GMW_LoadAPI("d:\\GM4", "d:\\GM4", "d:\\GM4\\demo",
 “*DDE_LOGIN_CREDENTIALS*”, szDDEReturnString);
```

Loading a BDE Session (GoldMine 6.7 or lower)

Syntax

C/C++	int GMW_LoadBDE(char *szSysDir, char *szGoldDir, char *szCommonDir, char *szUser, char *szPassword)
VB	Public Declare Function GMW_LoadBDE Lib "gm6s32.dll" (ByVal strSysDir As String, ByVal strGoldDir As String, ByVal strCommonDir As String, ByVal strUser As String, ByVal strPassword As String) As Long

Parameters

The GMW_LoadBDE function takes five parameters.

SzGoldDir: Specifies the location of CAL.DBF.

SzCommonDir: Specifies the location of CONTACT1.DBF.

SzUser: Specifies the GoldMine user name (must be UPPERCASE).
For API version 5.70.20222 and later: You may set this parameter to the value of *DDE_LOGIN_CREDENTIALS* to use login credentials returned for the user logged into a running copy of GoldMine through DDE.

SzPassword: Specifies the user’s password (must be UPPERCASE).
For API version 5.70.20222 and later: You may set this to the return string from the GetLoginCredentials DDE command if the User parameter is set to *DDE_Login_Credentials*. The credential string is only valid for 30 seconds.

Return Values

The GMW_LoadBDE function returns the following values:

GMW_LoadBDE Return Values

Return	Description
1	Success
0	BDE already loaded
-1	BDE failed to load
-2	Cannot find license file
-3	Cannot load license file
-4	Cannot validate the license file username/password
-5	Invalid GoldDir
-6	Invalid CommonDir
-7	Failed to allocate the needed TLS slot
-8	General Failure
-9	No access to specified contact set for this user

Notes

GMW_LoadBDE must be called before calling any function that accesses databases, such as GMW_UpdateSyncLog and GMW_ReadImpTLog. GMW_UnloadBDE must be called before unloading the DLL. GMW_LoadBDE may be called as many times as necessary. Be sure to match a corresponding GMW_UnloadBDE for every call of GMW_LoadBDE.

Example

```
GMW_LoadBDE( "d:\\GM4", "d:\\GM4", "d:\\GM4\\demo", "JON", “PASS” );
Or
GMW_LoadBDE("d:\\GM4", "d:\\GM4", "d:\\GM4\\demo",
 “*DDE_LOGIN_CREDENTIALS*”, szDDEReturnString);
```

Logging in a User

GMW_Login may be used to login a different user than was originally logged in through GMW_LoadAPI or GMW_LoadBDE.

Syntax

C/C++	int GMW_Login(char *szUser, char *szPassword, char *szSQLUser, char *szSQLPassword)
VB	Public Declare Function GMW_Login Lib "gm6s32.dll" (ByVal strUser As String, ByVal strPassword As String, Optional ByVal strSQLUser As String, Optional ByVal strSQLPassword As String) As Long

Parameters

szUser: Specifies the GoldMine user name (must be UPPERCASE).
For API version 5.70.20222 and later: You may set this parameter to the value of *DDE_LOGIN_CREDENTIALS* to use login credentials returned for the user logged into a running copy of GoldMine through DDE.

szPassword: Specifies the user's password (must be UPPERCASE).
For API version 5.70.20222 and later: You may set this to the return string from the GetLoginCredentials DDE command if the User parameter is set to *DDE_Login_Credentials*. The credential string is only valid for 30 seconds.

szSQLUser: Specifies the user's SQL login name. Omit if using DDE login credentials.

szSQLPassword: Specifies the user's SQL password. Omit if using DDE login credentials.

Return Values

The GMW_Login function returns the following values:

GMW_Login Return Values

Return	Description
1	Success
0	Failure
-1	User does not have permission to open the current contact set.

Example

```
GMW_Login( "JOE", "PASS", "SA", "");
Or
GMW_Login("*DDE_LOGIN_CREDENTIALS*", szDDEReturnString);
```

Closing an API Session (GoldMine 7.0 or higher)

Syntax

C/C++	int GMW_UnloadAPI()
VB	Public Declare Function GMW_UnloadAPI Lib "gm6s32.dll" () As Long

Return Values

The GMW_UnloadAPI function returns the following values:

GMW_UnloadBDE Return Values

Return	Description
0	Failure
1	Success

Notes

If GMW_LoadAPI is called, GMW_UnloadAPI must be called before unloading the DLL.

Example

GMW_UnloadAPI();

The following functions perform additional functions:

GMW_GetLicenseInfo: Returns GoldMine licensing information

Closing a BDE Session (GoldMine 6.7 or lower)

Syntax

C/C++	int GMW_UnloadBDE()
VB	Public Declare Function GMW_UnloadBDE Lib "gm6s32.dll" () As Long

Return Values

The GMW_UnloadBDE function returns the following values:

GMW_UnloadBDE Return Values

Return	Description
0	Failure
1	Success

Notes

If GMW_LoadBDE is called, GMW_UnloadBDE must be called before unloading the DLL.

Example

GMW_UnloadBDE();

The following functions perform additional functions:

GMW_SetSQLUserPass: Sets the SQL database login name and password

GMW_GetLicenseInfo: Returns GoldMine licensing information

Logging in Multiple Users through the API

Some integrated solutions for GoldMine require more than one user logged into GoldMine. These are usually some type of server application or a Web-based interface. The following functions enable you to handle these situations.

The first function call you will make will still be the GMW_LoadAPI or GMW_LoadBDE function. You must enter a valid username to call this function, but you can leave the password blank. You can also use *DDE_LOGIN_ CREDENTIALS* to call this function. For more information on the GMW_LoadAPI or GMW_LoadBDE functions, see and .

Logging In

To log in multiple users, use the GMW_MULogin function. Logging in a user with this function will use a seat of your GoldMine license.

Syntax

C/C++	int __stdcall GMW_MULogin (char* szUser, char* szPassword, char* szSQLUser, char* szSQLPassword, char* szCommonDir)
VB	Public Declare Function GMW_MULogin Lib "gm6s32.dll" (ByVal strUser As String, ByVal strPassword As String,ByVal strSQLUser As String, ByVal strSQLPassword As String, ByVal strCommonDir As String) As Long

Parameters

szUser is the GoldMine login name

szPassword is the GoldMine password

szSQLUser is the username for the MS SQL server

szSQLPassword is the password for the MS SQL server

szCommondir is to set a different, specific contact file directory for this user

Return Values

The GMW_MULogin function returns the following values:

GMW_MULogin Return Values

Return	Description
> 0	The session ID for this user
0	Failed to set TLS value
-1	Failed to load license file
-2	Failed to validate name and password
-3	No more seats available
-4	Unknown general exception
-5	User does not have access to the specified contact set.

Logging Out

To log out a user when multiple users are logged in, use the GMW_MULogout function. This function will free the license seat previously used by the GMW_MULogin function.

Syntax

C/C++	int __stdcall GMW_MULogout (int nSessionID)
VB	Public Declare Function GMW_MULogout Lib "gm6s32.dll" (ByVal nSessionID As Long) As Long

Parameters

nSessionID is the integer value returned by the GMW_MULogin function

Returns

The function will return TRUE if the specified SessionID was valid.

Switching Between Login Sessions

If you are working with more than one login session, it is important to note that the API functions always work on the last user logged in. The functions do not have a parameter to specify which session (user) to operate on. In order to switch to a different login session, use the GMW_MUBeginSession function.

Syntax

C/C++	int __stdcall GMW_MUBeginSession (int nSessionID)
VB	Public Declare Function GMW_MUBeginSession Lib "gm6s32.dll" (ByVal nSessionID As Long) As Long

Parameters

nSessionID is the integer value returned by the GMW_MULogin function and specifies which login session under which you want the API calls to operate.

Returns

The function returns the SessionID on success, and 0 on failure.

Special Consideration for Multi-Threaded Applications

There may be an instance when your application will not be able to guarantee that every data request will go through the same thread that created the session, such as the case with Internet Information Server. If you try to access an API session from a different thread than the one that created the session, you may encounter exceptions.

To handle these situations, use the GMXTP.DLL. Each of the functions in the GMXS32.DLL is wrapped through the GMXTP.DLL, so there is no need to load both. In addition, the above multiple login functions have slightly altered names:

GMW_TP_MULogin
GMW_TP_MULogout
GMW_TP_MUBeginSession

In addition, there is one additional function to be aware of, GMW_TP_CopySecurityTokentoWorkthread.

Syntax

C/C++	GMW_TP_CopySecurityTokentoWorkThread ()
VB	Public Declare Sub GMW_TP_CopySecurityTokentoWorkThread lib “gm6s32.dll” ()

This function ensures that the thread that is attempting access gets the identity of the working thread instead of the process. This function is especially important when dealing with IIS Extensions.

Working with Business Logic Functions using the Name/Value Pair Method

The following section describes the functions available for the programmer to manipulate Name Value containers, used for accessing the high-level business logic functions via the GMXS32.DLL. For function prototypes and code examples in C++, Visual Basic and Delphi, see .

For information on which business logic functions are available, and their expected name/value pairs, see ..

Notes

- These functions require that you are successfully logged into a GoldMine database using the GMW_LoadAPI or GMW_LoadBDE function.
- You must pass an empty NV container with all calls that do not take any parameters.

Creating an NV Container

GMW_NV_Create creates an NV container. This is the first step in using the name/value pair containers. This is analogous to creating a structure to store multiple variables indicating the values you wish to assign to fields in GoldMine.

Syntax

C/C++	HGMNV __stdcall GMW_NV_Create()
VB	Public Declare Function GMW_NV_Create Lib "gm6s32.dll" () As Long

Example

lGMNV = GMW_NV_Create

Return Value

Pointer to a new NV container

Creating an NV Container with Copied Values

GMW_NV_CreateCopy creates an NV container and copies the values from an existing NV container.

Syntax

C/C++	HGMNV __stdcall GMW_NV_CreateCopy(HGMNV hgmnv)
VB	Public Declare Function GMW_NV_CreateCopy Lib "gm6s32.dll" (ByVal hgmnv As Long) As Long

where hgmnv represents the pointer to the source NV container.

Example

lGMNV2 = GMW_NV_CreateCopy(pGMNV)

Return Value

Pointer to a new NV container.

Copying Values between NV Containers

GMW_NV_Copy copies the values from one NV container to another. GMW_NV_Create or GMW_NV_CreateCopy must have previously created both NV containers.

Syntax

C/C++	void _stdcall GMW_NV_Copy (HGMNV hgmnvDestination, HGMNV hgmnvSource)
VB	Public Declare Sub GMW_NV_Copy Lib "gm6s32.dll" (ByVal hgmnvDestination As Long, ByVal hgmnvSource As Long)

Parameters

hgmnvDestination is the pointer to the destination container.

hgmnvSource is the pointer to the source container.

Example

GMW_NV_Copy lGMNV2, lGMNV

Return Value

n/a

Deleting an NV Container

GMW_NV_Delete deletes an NV container and releases its memory. Be sure to call this for all previously created containers before exiting your application.

Syntax

C/C++	void __stdcall GMW_NV_Delete(HGMNV hgmnv)
VB	Public Declare Sub GMW_NV_Delete Lib "gm6s32.dll" (ByVal hgmnv As Long)

where hgmnv is the pointer to the NV container to delete.

Example

GMW_NV_Delete lGMNV

Return Value

n/a

Reading Values from an NV Container

GMW_NV_GetValue reads a value stored in an NV container. If the name does not exit in the container, the default value is returned. This method is used to read data out of the container returned from GoldMine. For example, after creating a contact, you would call GMW_NV_GetValue to read the new Recid or Accountno assigned to the contact.

Syntax

C/C++	const char* __stdcall GMW_NV_GetValue(HGMNV hgmnv, const char* name, const char* DefaultValue)
VB	Public Declare Function GMW_NV_GetValue Lib "gm6s32.dll" (ByVal hgmnv As Long, ByVal Name As String, ByVal DefaultValue As String) As GMWStr

Parameters

hgmnv is the pointer to a valid name value container

Name is the name of the value to return

DefaultValue is the default value if <Name> is null or does not exist.

Example

sValue = GMW_NV_GetValue (lGMNV, ‘Accountno’, ‘(none)’)

Return Values

The value of the Name is returned. If the Name is null or does not exist, the DefaultValue value is returned.

Storing NV Pairs in a Container

GMW_NV_SetValue stores a Name/Value pair in the specified container. Use this function to specify the values that you wish to assign to the GoldMine record (contact, cal, history, etc). Call this function for each field value you need to assign.

Syntax

C/C++	void __stdcall GMW_NV_SetValue(HGMNV hgmnv, const char* name, const char* value)
VB	Public Declare Sub GMW_NV_SetValue Lib "gm6s32.dll" (ByVal hgmnv As Long, ByVal Name As String, ByVal Value As String)

Parameters

hgmnv is the pointer to a valid name value container.

Name is the name of the value to set.

Value is the value to assign to <Name>.

Example

GMW_NV_SetValue lGMNV, 'Phone1', '(310)555-1212'

Return Value

n/a

Searching for an NV Pair

GMW_NV_NameExists checks if the specified Name/Value exists within the NV container.

Syntax

C/C++	long __stdcall GMW_NV_NameExists(HGMNV hgmnv, const char* name)
VB	Public Declare Function GMW_NV_NameExists Lib "gm6s32.dll" (ByVal hgmnv As Long, ByVal Name As String) As Long

Parameters

hgmnv is the pointer to a valid name value container.

Name is the name of the value to set.

Example

iResult = GMW_NV_NameExists (lGMNV, 'Phone1')

Return Values

GMW_NV_NameExists Return Values

Return	Description
0	Value does not exist in container
1	Value exists in container

Removing one NV Pair

GMW_NV_EraseName removes a Name/Value pair from the specified container. This function is useful for removing the Recid name/value pair from a container that has already been used once to create a new record. To reuse the container using all of the same name/value pairs, the Recid name/value pair needs to be removed in order to create another new record.

Syntax

C/C++	void __stdcall GMW_NV_EraseName(HGMNV hgmnv, const char* name)
VB	Public Declare Sub GMW_NV_EraseName Lib "gm6s32.dll" (ByVal hgmnv As Long, ByVal Name As String)

Parameters

hgmnv is the pointer to a valid name value container

Name is the name of the value to set

Example

GMW_NV_EraseName lGMNV, ‘Phone1’

Return Value

n/a

Removing all NV Pairs from a Container

GMW_NV_EraseAll removes all Name/Value pairs from the specified container.

Syntax

C/C++	void __stdcall GMW_NV_EraseAll(HGMNV hgmnv)
VB	Public Declare Sub GMW_NV_EraseAll Lib "gm6s32.dll" (ByVal hgmnv As Long)

Parameter

hgmnv is the pointer to a valid name value container.

Example

GMW_NV_EraseAll lGMNV

Return Value

n/a

Totaling NV Pairs in a Container

GMW_NV_Count returns the number of Name/Value pairs within the specified container.

Syntax

C/C++	long __stdcall GMW_NV_Count(HGMNV hgmnv)
VB	Public Declare Function GMW_NV_Count Lib "gm6s32.dll" (ByVal hgmnv As Long) As Long

Parameter

hgmnv is the pointer to a valid name value container.

Example

```
iCount = GMW_NV_Count lGMNV
```

Return Value

Number of NVs within the specified container.

Finding an NV Name

GMW_NV_GetNameFromIndex finds the name of the NV stored at a specific index within the container. The first item in the container is at index value 1.

Syntax

C/C++	const char* __stdcall GMW_NV_GetNameFromIndex(HGMNV hgmnv, long index))
VB	Public Declare Function GMW_NV_GetNameFromIndex Lib "gm6s32.dll" (ByVal hgmnv As Long, ByVal index As Long) As GMWStr

Parameters

hgmnv is the pointer to a valid name value container

Index is the item number to return.

Example

```
sName = GMW_NV_GetNameFromIndex(lGMNV, 3)
```

Return Value

The name stored at <Index> within the container.

Finding an NV Value

GMW_NV_GetValueFromIndex finds and returns the value of the NV stored at the specified index within the container. The first item in the container is stored an index value 1.

Syntax

C/C++	const char* __stdcall GMW_NV_GetValueFromIndex(HGMNV hgmnv, long index)
VB	Public Declare Function GMW_NV_GetValueFromIndex Lib "gm6s32.dll" (ByVal hgmnv As Long, ByVal index As Long) As GMWStr

Parameters

hgmnv is the pointer to a valid name value container

Index is the item number to return

Example

```
sValue = GMW_NV_GetValueFromIndex(pGMNV, 3)
```

Return Value

The value stored at <Index> within the container.

Setting NV Pairs

GMW_NV_SetStr sets one or more Name/Value pairs. This function is used if you would like to set multiple name/value pairs in a single call.

Syntax

C/C++	void __stdcall GMW_NV_SetStr(HGMNV hgmnv, char dlmName, char dlmVal, const char* pszValueStr)
VB	Public Declare Sub GMW_NV_SetStr Lib "gm6s32.dll" (ByVal hgmnv As Long, ByVal strDlmName As String, ByVal strDlmVal As String, ByVal ValueStr As String)

Parameters

hgmnv is the pointer to a valid name value container.

DlmName is the delimiter between the name and its value.

DlmVal is the delimiter between each NV pairs.

ValueStr is the string containing the name values.

Example

```
GMW_NV_SetStr lGMNV,'=',';','Company=GoldMine;Key1=Cust'
GMW_NV_SetStr lGMNV,'&','&','Company&GoldMine&Key1&Cust'
```

NOTE: * The delimiters may be the same for *DlmName* and *DlmVal*.

Return Value

n/a

Executing Business Logic Methods

All of the Business Logic methods are accessed through the GMW_Execute function. You must be successfully logged into a GoldMine database for this call to work properly. For details about Business Logic methods, see .

Syntax

C/C++	long _stdcall GMW_Execute(const char *szFuncName, HGMNV hgmnv)
VB	Public Declare Function GMW_Execute Lib "gm6s32.dll" (ByVal strFuncName As String, ByVal hgmnv As Any) As Long

Parameters

FuncName is one of the various business logic functions described below.

hgmnv is the pointer to a Name/Value container.

Example

```
GMW_Execute "WriteContact", lGMNV
```

Return Values

GMW_Execute Return Values

Return	Description
0	Failure
>0	Success

Working with Multi-Value Name/Value Pairs

Some business logic methods use a special name/value pair that contains multiple values. In addition, a name/value pair may simply hold a string value, or it may hold the handle(s) to one or more name/value containers. The lifetime of an embedded NV value is controlled by its parent. You do not need to call GMW_NV_Delete on it.

The following functions are used to manipulate and read multi-value pairs.

Determining the Type of a Name/Value Pair

The GMW_NV_GetValueType function is used to determine if a name/value pair is a multi-value pair or a container.

GoldMine API Version: 5.50.10111

Syntax

C/C++	long _stdcall GMW_NV_GetValueType(HGMNV hgmnv, const char *name)
VB	Public Declare Function GMW_NV_GetValueType Lib "gm6s32.dll" (ByVal hgmnv As Long, ByVal strName As String) As GMWNVValueType

Parameters

hgmnv is the pointer to a Name/Value container.

Name is the name of the name/value pair for which you want to determine the type.

Return Values

Possible return values are as follows:

GetValueType Return Values

Value	Description
GM_NV_VALUE_TYPE_SINGLE_NV	The value is one NV Containers
GMW_NV_VALUE_TYPE_MULTI_NV	The value stores multiple NV containers
GMW_NV_VALUE_TYPE_MULTI_STRING	The value stores multiple string values

Determining the Position of an NV Container in an NV Hierarchy

If the value in an NV pair contains another container, the container that holds the second container is the parent of the second container. When there are no more parents, or you are at the top level of the hierarchy, the container is considered the root. The following functions will indicate whether the container is a parent or root, or return the handle to the root or parent.

GoldMine API Version: 5.50.10111

Syntax

C/C++	BOOL _stdcall GMW_NV_IsRoot(HGMNV hgmnv)
VB	Public Declare Function GMW_NV_IsRoot Lib "gm6s32.dll" (ByVal hgmnv As Long) As Long

Returns TRUE (not zero) if the specified hgmnv is the root.

Parameters

hgmnv is the pointer to a Name/Value container.

Example

If(GMW_NV_is Root (hgmnv)) {it’s the root} else {it’s a child}

Syntax

C/C++	HGMNV _stdcall GMW_NV_GetRoot(HGMNV hgmnv)
VB	Public Declare Function GMW_NV_GetRoot Lib "gm6s32.dll" (ByVal hgmnv As Long) As Long

Returns the hgmnv of the root for the specified container. If the root's hgmnv is specified, the same hgmnv will be returned.

Parameters

hgmnv is the pointer to a Name/Value container.

Example

```
hRootNV = GMN_NV_GetRoot(hgmnv)
```

Syntax

C/C++	HGMNV _stdcall GMW_NV_GetParent(HGMNV hgmnv)
VB	Public Declare Function GMW_NV_GetParent Lib "gm6s32.dll" (ByVal hgmnv As Long) As Long

Returns the hgmnv of the parent for the specified container. The function returns NULL if the specified hgmnv has no parent (is the root).

Parameters

hgmnv is the pointer to a Name/Value container.

Example

```
hParentNV = GMW_NV_GetParent(hgmnv)
```

Getting the Number of Values in a Multi-Value Pair

The GMW_NV_GetMultiValueCount function will return the number of values included in a multi-value name/value pair.

GoldMine API Version: 5.50.10111

Syntax

C/C++	long __stdcall GMW_NV_GetMultiValueCount(HGMNV hgmnv, const char* name)
VB	Public Declare Function GMW_NV_GetMultiValueCount Lib "gm6s32.dll" (ByVal hgmnv As Long, ByVal strName As String) As Long

Parameters

hgmnv is the pointer to a Name/Value container.

Name is the name of the name/value pair for which you want to receive the count of values.

Example

```
numberOfValues = GMW_NV_GetMultiValueCount(hgmnv, “POP3_Account”)
```

Retrieving Containers from an NV Pair

When a value contains one container, the GMW_NV_GetNVValue function is used to retrieve the hgmnv for that child container.

GoldMine API Version: 5.50.10111

Syntax

C/C++	HGMNV _stdcall GMW_NV_GetNvValue(HGMNV hgmnv, const char* name)
VB	Public Declare Function GMW_NV_GetNvValue Lib "gm6s32.dll" (ByVal hgmnv As Long, ByVal strName As String) As Long

Parameters

hgmnv is the pointer to a Name/Value container.

Name is the name of the name/value pair from which you want to receive the child container.

Example

```
hSubNV = GMW_NV_GetNvValue(hgmnv, “TheNVName”)
```

When a value contains multiple containers, the GMW_NV_GetMultiNvValue function is used to retrieve the hgmnv for the child containers.

Syntax

C/C++	HGMNV _stdcall GMW_NV_GetMultiNvValue(HGMNV hgmnv, const char* name, long position);
VB	Public Declare Function GMW_NV_GetMultiNvValue Lib "gm6s32.dll" (ByVal hgmnv As Long, ByVal strName As String, ByVal position As Long) As Long '1 based

Parameters

hgmnv is the pointer to a Name/Value container.

Name is the name of the name/value pair from which you want to receive the child container.

Position is the nth value you want to retrieve (1 based). If you wanted the tenth container in the value, then position would be 10.

Example

```
hSubNV = GMW_NV_GtMultiNvValue(hgmnv, “TheNVName”, 10)
```

Retrieving the Values in a Multi-Value Pair

The GMW_NV_GetMultiValue function is used to retrieve the values from a multi-value pair. It is called for each value and the number of the value to retrieve must be specified. This function is used to retrieve string values. To retrieve NV containers from the value, use the GMW_NV_GetNvValue function or the GMW_NV_GetMultiNvValue function.

GoldMine API Version: 5.50.10111

Syntax

C/C++	const char* _stdcall GMW_NV_GetMultiValue(HGMNV hgmnv, const char* name, long element, const char* defaultValue)
VB	Public Declare Function GMW_NV_GetMultiValue Lib "gm6s32.dll" (ByVal hgmnv As Long, ByVal strName As String, element As Long, ByVal strDefaultValue As String) As GMWStr

Parameters

hgmnv is the pointer to a Name/Value container.

Name is the name of the name/value pair for which you want to receive the values from.

Element is the number of the value to be returned. This is 1 based.

DefaultValue is the default value to return if the element supplied is not found.

Example

To return the fifth element:

```
strFifthElemnt = GMW_NV_GetMultiValue(hgmnv,
“POP3_Account”, 5, “No Account”)
```

Deleting Values from a Multi-Value Pair

The GMW_NV_EraseName function will delete the entire Multi-Value Pair.

GoldMine API Version: 5.50.10111

Assigning a Container to a Parent

If you need to populate a container that will be a child container, one approach is to create the container, fill int with its respective values, and then copy the container into the value of the NV pair desired.

When the NV pair holds only one container, the GMW_NV_SetNvValue function is used.

GoldMine API Version: 5.50.10111

Syntax

C/C++	void _stdcall GMW_NV_SetNvValue(HGMNV hgmnv, const char* name, HGMNV hgmnvValue)
VB	Public Declare Sub GMW_NV_SetNvValue Lib "gm6s32.dll" (ByVal hgmnv As Long, ByVal strName As String, ByVal hgmnvValue As Long)

Parameters

hgmnv is the pointer to the parent Name/Value container.

Name is the name of the name/value pair into which you want to copy the child container.

hgmnvValue is the prepared NV container to copy to the parent container.

Example

```
GMW_NV_SetNvValue hgmnv, "TheNVName", hChildNV
```

The GMW_NV_AppendNvValue function will append a copy of the specified child container to an NV pair value that contains multiple containers.

Syntax

C/C++	long _stdcall GMW_NV_AppendNvValue(HGMNV hgmnv, const char* name, HGMNV hgmnvValue)
VB	Public Declare Function GMW_NV_AppendNvValue Lib "gm6s32.dll" (ByVal hgmnv As Long, ByVal strName As String, ByVal hgmnvValue As Long) As Long

Parameters

hgmnv is the pointer to the Name/Value container.

Name is the name of the name/value pair into which you want to copy the child container.

hgmnvValue is the prepared NV container to copy to the parent container.

Example

```
GMW_NV_AppendNvValue hgmnv, "The NVName", hChildNV
```

Creating an Empty Child Container Within the Parent

The two preceding functions took a prepared NV container and copied it to the parent container. Another (best practice) method would be to allow the API to create the child container for you, return the hgmnv to that child, and then allow you to fill it with the appropriate values.

The GMW_NV_SetEmptyNvValue will create a child container for an NV pair and return the hgmnv for that child. This function is used when the value is to hold only one child container.

GoldMine API Version: 5.50.10111

Syntax

C/C++	HGMNV _stdcall GMW_NV_SetEmptyNvValue(HGMNV hgmnv, const char* name)
VB	Public Declare Function GMW_NV_SetEmptyNvValue Lib "gm6s32.dll" (ByVal hgmnv As Long, ByVal strName As String) As Long

Parameters

hgmnv is the pointer to the parent Name/Value container.

Name is the name of the name/value pair in which you want to create the child container.

Example

```
hChildNv = GMW_NVSetEmptyNvValue(hgmnv, "TheNVName")
'now set the values of the child container using the returned HGMNV
```

When you need to append an empty child container to an NV pair containing multiple children, use the GMW_NV_AppencdEmptyNvValue function.

Syntax

C/C++	HGMNV _stdcall GMW_NV_AppendEmptyNvValue(HGMNV hgmnv, const char* name)
VB	Public Declare Function GMW_NV_AppendEmptyNvValue Lib "gm6s32.dll" (ByVal hgmnv As Long, ByVal strName As String) As Long

Parameters

hgmnv is the pointer to the parent Name/Value container.

Name is the name of the name/value pair to which you want to append the new empty child container.

Example

```
hChildNv = GMW_NV_AppendEmptyNvValue(hgmnv, "TheNVName")
'now set the values of the child container using the returned HGMNV.
```

Appending String Values to a Multi-Value Pair

The GMW_NV_AppendValue function will append values to a multi-value pair.

GoldMine API Version: 5.50.10111

Syntax

C/C++	long _stdcall GMW_NV_AppendValue(HGMNV hgmnv, const char* name, const char* value)
VB	Public Declare Function GMW_NV_AppendValue Lib "gm6s32.dll" (ByVal hgmnv As Long, ByVal strName As String, ByVal strValue As String) As Long

Parameters

hgmnv is the pointer to a Name/Value container.

Name is the name of the name/value pair for which you want to receive the count of values.

Value is the value to be appended to the end of the list of values.

Example

To set five (5) values for the POP3_Account value:

```
For i = 1 To 5
GMW_NV_Append hgmnv, "POP3_Account", i
Next i
```

Low-level Data Access & Manipulation

The following sections describe additional functions in the `GMXS32.DLL` that allow data reading and updating via low-level methods. Use of the following functions requires in-depth knowledge of the GoldMine data structures and business rules. They are useful for accessing and writing data that is not accessible via the high-level business logic functions.

Reading Security and Rights for a DLL User

The GMW_UserAccess function retrieves specific permission information for the logged-in user.

GoldMine API Version: 5.00.041

Syntax

C/C++	int _stdcall GMW_UserAccess(long iOption)
VB	Public Declare Function GMW_UserAccess lib "gm6s32.dll" (ByVal iOption as long) as Integer

Parameters

GMW_UserAccess takes one parameter, iOption, which is a value for the types of rights settings you wish to query.

iOption values

Value	Rights
100	Master Rights
101	Access to other user's calendar
102	Access to other user's history
103	Access to other user's sales

104	Access to other user's reports
105	Access to other user's merge forms
106	Access to other user's filters
107	Access to other user's groups
108	Access to other user's links
111	Right to create a record
112	Right to edit a record
113	Right to delete a record
114	Right to change record owner
115	Right to field views
116	Right to schedule automated processes
118	Right to SQL Query
119	Right to NetUpdate
124	Right to build groups

Return Values

The GMW_UserAccess function returns 1 if the user has the queried rights.

Using GMW_CalAccess, you can query whether the user logged in via the DLL has rights to read/write a CAL record.

Syntax

C/C++	int _stdcall GMW_CalAccess(char *szRecType, char *szUserID, char *szNumber1)
VB	Public Declare Function GMW_CalAccess lib "gm6s32.dll" (ByVal sRectype as String, ByVal sUserID as String, ByVal sNumber1 as String) as Integer

Parameters

szRecType is the RecType of the record.

szUserID is the UserID of the record.

szNumber1 is the Number1 value of the record.

Return Values

The GMW_CalAccess function returns 1 if the user has rights to read/write.

Using GMW_HistAccess, you can query if the user logged in via the DLL has rights to read/write a CONTHIST record.

Syntax

C/C++	int _stdcall GMW_HistAccess(char *szRecType, char *szUserID)
VB	Public Declare Function GMW_HistAccess Lib "gm5s32.dll" (ByVal szRecType As String, ByVal szUserID As String) As Integer

Parameters

szRecType is the RecType of the record.

szUserID is the UserID of the record.

Return Values

The GMW_HistAccess function returns **1** if the user has rights to read/write.

Returning GoldMine Licensing Information

GoldMine API Version: 5.00.041

Syntax

C/C++	int_stdcall GMW_GetLicenseInfo(GMW_LicInfo *pLic)
VB	Public Declare Function GMW_GetLicenseInfo Lib "gm6s32.dll" (LicInfo As GMW_LicInfo) As Long

Parameters

GMW_GetLicenseInfo takes one parameter pLic, which is a pointer to a client allocated GMW_LicInfo structure.

Return Values

The GMW_GetLicenseInfo function returns the following values:

GMW_GetLicenseInfo Return Values

Return	Description
0	Failure
1	Success

Notes

The GMW_LicInfo structure includes the following items:

GMW_GetLicenseInfo Structure

Type/Size	Name	Description

char / 60	Licensee	Licensee name
char / 40	LicNo	Master serial number
char / 20	SiteName	Undocked site name
long integer	LicUsers;	Licensed users
long integer	SQLUsers;	Licensed SQL users
long integer	GSSites;	License GoldSync sites
long integer	isDemo;	Is demo install? 1=True
long integer	isServerLic;	Is primary ('D' or 'E') license? 1=True
long integer	isRemoteLic;	Is remote (‘U’ or ‘S’) license? 1=True
long integer	isUSALicense;	Is USA license? 1=True
long integer	DLLVersion	DLL Version number
long integer	Reserved1	Reserved
long integer	Reserved2	Reserved
char / 100	sReserved	Reserved

Example

```
GMW_LicInfo oLic;
GMW_GetLicenseInfo( &oLic;
```

Returning Calendar Data

The ReadSchedule call returns all calendar data for a given RecID. You can also make the ReadSchedule call through the XML API.

Syntax

C/C++	pnv = (GMWnv*)GMW_NV_CreateCls();
	pnv->Set("RecID", "SOMEVALIDRECID");
	GMW_NV_Execute("ReadSchedule", pnv);

Retrieving Data with DataStream

DataStream returns the data of ordered records from any GoldMine table using the most efficient method available. The caller can specify:

- Fields and expressions to return
- Range of records to return
- Optional filter to apply to the data set

DataStream SQL query capabilities are very fast on SQL databases.

The DataStream method allows for many useful applications. One such group of applications would merge HTML templates with the data returned by GoldMine DataStream to publish the contents of GoldMine data on the Internet. Web pages can be created to display GoldMine data requested by a visitor. Based on visitor selections, a company could dynamically present a variety of HTML pages, including dealer addresses in a particular city, financial numbers stored in Contact2, and even seating availability at upcoming conferences. With a fast Internet connection and a strong SQL server, the GoldMine client could respond simultaneously to dozens of requests.

Advantages of Using DataStream

GoldMine DataStream is absolutely the fastest way to read data from GoldMine tables. Used correctly, DataStream will return the data faster than most development environments would directly. DataStream offers the following advantages:

- *Efficiency*: DataStream issues a single, most efficient SQL query or Xbase seek to retrieve records from the back-end database to the local client. On SQL databases, requests of a few hundred records could be sent from the server to the client with a single network transaction, greatly minimizing network traffic.
- *Speed*: All fields and expressions are parsed initially by GMW_DS_Range() and GMW_DS_Query(), and then quickly evaluated against each record in GMW_DS_Fetch. Other DDE methods (and development environments) require that each field be parsed and evaluated each time its data is read. This makes a big difference when reading hundreds or thousands of records.
- *Simplicity*: Only three function calls are required to read all the data. Using traditional record-by-record querying would require one call for each field of each record (reading 10 fields from 50 records would require 500 function calls).
- *Results*: All the work to gather and format the data is done in C++, which is the fastest way to fly. The caller needs only to parse the resulting packet string.

DataStream Record Selection

The following DataStream functions are listed in the order in which they must be called.

GMW_DS_Range(): Opens a ranged cursor

GMW_DS_Query(): Opens an SQL query cursor

GMW_DS_Fetch(): Fetches records

GMW_DS_Close(): Closes cursor

Either the GMW_DS_Range() function or the GMW_DS_Query() function must be called first to request the data. These functions return the integer handle, iHandle, which must be passed to the GMW_DS_Fetch() and GMW_DS_Close() functions.

You must use either GMW_DS_Range() or GMW_DS_Query()—you cannot use both. The GMW_DS_Range and GMW_DS_Query functions execute equally fast on SQL and FireBird databases. GMW_DS_Range executes much faster on Xbase tables than does GMW_DS_Query.

GMW_DS_Range

Syntax

C/C++	long GMW_DS_Range(char *szTable, char *szTag, char *szTopLimit, char *szBotLimit, char *szFields, char *szFilter, char *szFDlm, char *szRDlm);
VB	Public Declare Function GMW_DS_Range Lib "gm6s32.dll" (ByVal strTable As String, ByVal strTag As String, ByVal strTopLimit As String, ByVal strBotLimit As String, ByVal strFields As String, ByVal strFilter As String, ByVal strFDlm As String, ByVal strRDlm As String) As Long

GMW_DS_Range returns a range of records based on an index.

Parameters

The following parameters are required:

szTable specifies the table name (such as “Contact1”) or the table ID.

szTag designates the tag that corresponds to the index file.

szTopLimit specifies the top limit of the range. (Must conform to the index expression.)

szBotLimit specifies the bottom limit of the range. (Must conform to the index expression.)

szFields specifies the requested fields and expression to return—see “GMW_DS_Range Field Selection” on the following page.

The following parameters are optional:

szFilter designates an optional Xbase filter expression.

szFDlm specifies the field delimiter (default: carriage return).

szRDlm specifies the record delimiter (default: line feed).

Return Values

The GMW_DS_Range function returns the following values:

GMW_DS_Range Return Values

Return	Description
0	Failure
1–20	Success (handle)

GMW_DS_Range Field Selection

The szFields parameter passed to GMW_DS_Range should consist of the field names and Xbase expressions to evaluate against each record in the data set. Each field must be terminated with a semicolon (;). Xbase expressions must be prefixed with an ampersand (&), and terminated with a semicolon. For example, the following commands request the first 100 cities from the Lookup file, including the city name and record number (RecID under SQL):

```
ih1 = GMW_DS_Range( "lookup", "lookup", "CITY", "CITYZ", "Entry; &RecNo
();")
r1 = GMW_DS_Fetch( ih1, szBuf, iBufSize, 100 )
r2 = GMW_DS_Close( ih1 )
```

The following commands request the first 10 profiles of the current contact record, followed by a request for the next 50 profiles:

```
ih1 = GMW_DS_Range( "contsupp","contspfd", sAccNo+"P", sAccNo+"P",
"Contact;ContSupRef;")
r1 = GMW_DS_Fetch( ih1, szBuf, iBufSize, 10 )
r1 = GMW_DS_Fetch( ih1, szBuf, iBufSize, 50 )
r1 = GMW_DS_Close( ih1 )
```

GMW_DS_Query

Syntax

C/C++	long GMW_DS_Query(char *szSQL, char *szFilter, char *szFDlm, char *szRDlm);
VB	Public Declare Function GMW_DS_Query Lib "gm6s32.dll" (ByVal strSQL As String, Optional ByVal strFilter As String, Optional ByVal strFDlm As String, Optional ByVal strRDlm As String) As Long

This function is very fast on SQL databases.

Parameters

szSQL query sends the query for evaluation on the server. The SQL query can join multiple tables and return any number of fields.

Optional parameter **szFilter** specifies a Boolean Xbase filter expression to apply to the data set (even on SQL tables), similar to the DDE SETFILTER command.

Optional parameter **szFDlm** overrides the return packet's default field delimiter of CR (carriage return).

Optional parameter **szRDlm** overrides the return packet's default record delimiter of LF (line feed).

Return Values

The GMW_DS_Query function returns the following values:

GMW_DS_QueryReturn Values

Return	Description
0	Failure
-1	Invalid Query/Timeout
1–20	Success (handle)

GMW_DS_Fetch

Syntax

C/C++	long GMW_DS_Fetch(long iHandle, char *szBuf, int iBufSize, int nGetRecs);
VB	Public Declare Function GMW_DS_Fetch Lib "gm6s32.dll" (ByVal iHandle As Long, ByVal strbuf As String, ByVal iBufSize As Long, ByVal nGetRecs As Long) As Long

GMW_DS_Fetch returns a single packet string containing the requested data from all records processed by the current "fetch" command, as specified by the nGetRecs parameter. iHandle must be the value returned from GMW_DS_Range() or GMW_DS_Query(). For details about the packet format, see below.

GMW_DS_Fetch Return Packet

GMW_DS_Fetch returns a single packet string containing the data from all requested records. The packet includes a header record, followed by one record for each record evaluated by "fetch." Within each record in the packet, the fields are separated by a field delimiter specified in GMW_DS_Range or GMW_DS_Query. By default, the field delimiter is the carriage return character (13 or 0x0D).

The records in the packet are separated by the record delimiter. By default, the record delimiter is the line feed character by default (10 or 0x0A).

These delimiters are convenient when the requested data does not contain notes from blob fields. You can pass 0 for szFDlm, szRDlm to use the default delimiters. When requesting notes, override the default delimiters by passing other delimiter values to GMW_DS_Range() and GMW_DS_Query(). For packets with notes, good delimiters are the ASCII characters 1 and 2.

The City Lookup example from above might return a packet of data similar to:

```
3000-0004
Boston|23
London|393
Los Angeles|633
New York|29
```

The packet header record consists of two sections:

First byte can be 0, 3, or 4:

0 indicates that more records are available, which could be fetched with another GMW_DS_Fetch call

3 indicates the end-of-file (EOF)

4 indicates the beginning-of-file (BOF)

Number following the dash indicates the total number of data records contained in the packet.

DataStream takes about as much time to read three records as to read 30. For best performance, adjust the number of records requested by GMW_DS_Fetch to return 8K–32K packets.

The calling application must allocate the memory for a large enough packet buffer, and pass that memory buffer to GMW_DS_Fetch. When the number of records cannot be estimated to allocate a packet buffer, GMW_DS_Fetch can be called twice, once to fetch the data and return a buffer size, and a second time to retrieve the data into the buffer. When GMW_DS_Fetch is first called to get the buffer size, the szBuf and iBufSize parameters must both be 0. The nGetRecs parameter must indicate the number of records to fetch. When GMW_DS_Fetch is then called to retrieve the data that has been fetched by the first call, the nGetRecs parameter must be 0.

NOTE: If the return DataStream is too large for the specified buffer size, GMW_DS_Fetch returns a value of -*5*. When the buffer in increased to an adequate size, GMW_DS_Fetch will return the data in a DataStream. This behavior prevents the dropping of data due to undersized buffers.

GMW_DS_Close

Syntax

C/C++	long GMW_DS_Close(long iHandle)
VB	Public Declare Function GMW_DS_Close Lib "`gm6s32.dll`" (ByVal iHandle As Long) As Long

GMW_DS_Close must be called when the operation is complete. Unclosed data streams will leak memory and leave the database connections needlessly open. Passing an iHandle of 0 closes all open DataStream objects.

Accessing Low-Level Data Using Work Areas

The GoldMine `GMXS32.DLL` provides a complete set of functions that allow low-level access to the database tables. Using these functions, you can:

- Open particular data files
- Seek the values of the fields in the records in the data files
- Append records to the tables
- Delete records
- Replace data in the records

Database applications that need varied access to GoldMine data typically use this suite of functions. To work successfully, these functions rely on a work area parameter. Using this parameter, you can open multiple data files concurrently and manipulate each file independently by referencing the file by work area. These functions also maintain synchronization information, which is stored in the TLogs.

`GMXS32.DLL` offers the low-level access functions that are listed in the following table.

GMXS32.DLL Low-Level Access Functions

Function Name	Description
Opening and Closing Databases	
GMW_DB_Open	Opens one GoldMine data file for processing by another application

GMW_DB_Close	Releases a previously OPENed file when processing is complete
GMW_DB_IsSQL	In GM 7.0, Determines whether the table is MSSQL (1) or Other (0). Use the getDBEngineType function to retrieve additional DB engine information.
Creating and Deleting Records	
GMW_DB_Append	Adds a new, empty record to a GoldMine data file
GMW_DB_Delete	Deletes the current record in the specified work area.
Reading and Writing Data	
GMW_DB_Read	Queries a data file for the value of a field
GMW_DB_RecNo	Determines either current record number position (Xbase), or the record ID (SQL)
GMW_DB_Replace	Changes the value in a particular field in one GoldMine data file
GMW_DB_Unlock	Unlocks a record previously locked by a call to either GMW_DB_Append or GMW_DB_Replace
Limiting Scope of Data	
GMW_DB_Filter	Limits access to data in a GoldMine database by creating a subset of records based on expression criteria
GMW_DB_Range	Activates the index in a table, and sets a range of values to limit the scope of data that GoldMine will search
Searching for Data	
GMW_DB_Search	Performs a sequential search on a file
GMW_DB_Seek	Positions to the first record matching the seek value
GMW_DB_SetOrder	Sets the current index tag on the table
Navigating the Database	
GMW_DB_Move	Positions the record pointer to a particular record in a data file
GMW_DB_Goto	Positions to a specific record in the table
GMW_DB_Top	Positions to the first record in the table
GMW_DB_Skip	Positions to the next or prior record in the table
GMW_DB_Bottom	Positions to the last record in the table

GMXS32.DLL Low-Level Access Functions

Function Name	Description
GMW_DB_QuickSeek	Wraps several DLL functions to perform a Seek based on an index
GMW_DB_QuickRead	Wraps several DLL function to perform a Read
GMW_DB_QuickReplace	Wraps several DLL functions to perform a Replace

Detailed descriptions of each database access function appear on the following pages. Some of the following functions refer to table names, field names, and index tags. For details, see or .

Opening a Data File

GMW_DB_Open opens one GoldMine data file for processing by another application.

Syntax

C/C++	long GMW_DB_Open(char *szTablename);
VB	Public Declare Function GMW_DB_Open Lib "gm6s32.dll" (ByVal strTableName As String) As Long

Parameter

The GMW_DB_Open function takes only szTableName, which is the name of the table to be opened.

Return Values

The GMW_DB_Open function returns the following values:

GMW_DB_Open Return Values

Return	Description
0	Error occurred
>0	Work area handle for table

Closing a Data File

GMW_DB_Close releases a previously OPENed file when processing is complete. All previously opened files must be properly closed—failure to do so can result in database errors.

Syntax

C/C++	long GMW_DB_Close(long pArea);
VB	Public Declare Function GMW_DB_Close Lib "gm6s32.dll" (ByVal lArea As Long) As Long

Parameters

The GMW_DB_Close function takes only pArea, which is the work area handle of the file opened by the GMW_DB_Open function.

Return Values

The GMW_DB_Close function returns the following values:

GMW_DB_Close Return Values

Return	Description
0	Error occurred
1	Table properly closed

Checking for an SQL Table

GMW_DB_IsSQL is used to determine if the table is MSSQL (1) or Other (0). Use the *getDBEngineType* function to retrieve more detailed DB engine information.

Syntax

C/C++	long GMW_DB_IsSql(long pArea);
VB	Public Declare Function GMW_DB_IsSQL Lib "gm6s32.dll" (ByVal lArea As Long) As Long

Parameter

The GMW_DB_IsSQL function takes only pArea, which is the work area handle of the file opened by the GMW_DB_Open function.

Return Values

The GMW_DB_IsSQL function returns the following values in GoldMine 7.0:

GMW_DB_IsSQL Return Values

Return	**Description**
0	Table is not MSSQL
1	Table is MSSQL

Adding a Record

GMW_DB_Append adds an empty record to a GoldMine data file.

Syntax

C/C++	long GMW_DB_Append(long pArea, char* szRecID);
VB	Public Declare Function GMW_DB_Append Lib "gm6s32.dll" (ByVal lArea As Long, ByVal strRecID As String) As Long

Before using GMW_DB_Append, you must open a data file using the GMW_DB_Open function. After executing the GMW_DB_Append function, the record pointer is positioned at the new empty record, and the record is locked and ready to accept field replacements.

When a CONTACT1 record is appended, GoldMine automatically fills in the new record with the appropriate ACCOUNTNO and CREATEBY values. For all other records, you must replace the ACCOUNTNO field with the value from the CONTACT1 record with which the new record is to be linked. For records that require remote synchronization initialization, GoldMine will automatically fill in the value of the RECID field when these records are appended.

Parameters

pArea is the work area handle of the file opened by the GMW_DB_Open function.

szRecID specifies the size of the character buffer to accept the return value. The szRecID buffer must be at least 20 characters.

Return Value

Xbase: APPEND function returns the record number of the new record, or 0 if the file could not be locked.

SQL and FireBird: APPEND function returns the RECID of the new record in the szRecID parameter.

Deleting the Current Record

GMW_DB_Delete deletes the current record in the specified work area and moves the record pointer to the next record.

For records that require remote synchronization initialization, GoldMine will automatically maintain the TLog entry.

Syntax

C/C++	long GMW_DB_Delete(long pArea);
VB	Public Declare Function GMW_DB_Delete Lib "gm6s32.dll" (ByVal lArea As Long) As Long

Parameter

The GMW_DB_Delete function takes only pArea, which is the work area handle of the file opened by the GMW_DB_Open function.

Return Values

The GMW_DB_Delete function returns the following values:

GMW_DB_Delete Return Values

Return	Description
0	Error occurred
1	Record deleted

Querying for a Field Value

GMW_DB_Read queries a data file for the value of a field.

Syntax

C/C++	long GMW_DB_Read(long pArea, char *szField, char *szBuf, int iBufSize);
VB	Public Declare Function GMW_DB_Read Lib "gm6s32.dll" (ByVal lArea As Long, ByVal strField As String, ByVal strbuf As String, ByVal lBufSize As Long) As Long

Parameters

pArea is the work area handle of the file opened by the GMW_DB_Open function.

szField is the name of the field to read within the table.

szBuf is the buffer in which the function will return the results.

iBufSize specifies the size of the buffer.

GMW_DB_Range Return Values

Return	Description
0	Error occurred
1	Success

Checking the Current Record Number or Record ID

GMW_DB_RecNo is used to determine either current record number position (Xbase) or the record ID (SQL and FireBird).

Syntax

C/C++	long GMW_DB_RecNo(long pArea, char *szRecID);
VB	Public Declare Function GMW_DB_RecNo Lib "gm6s32.dll" (ByVal lArea As Long, ByVal strRecID As String) As Long

Parameters

pArea is the work area handle of the file opened by the GMW_DB_Open function.

SzRecID is a character string that accepts the return value of RecNo (Xbase) or RecID (SQL).

Return Value

Xbase: Returns the current record number

SQL: Returns the current RecID

Changing a Field Value

GMW_DB_Replace changes the value in a particular field in one GoldMine data file.

For records that require remote synchronization initialization, GoldMine will automatically maintain the TLog entry.

Syntax

C/C++	long GMW_DB_Replace(long pArea, char *szField, char *szData, int iAddTo);
VB	Public Declare Function GMW_DB_Replace Lib "gm6s32.dll" (ByVal lArea As Long, ByVal strField As String, ByVal strData As String, ByVal iAddTo As Long) As Long

Parameters

pArea is the work area handle of the file opened by the GMW_DB_Open function.

szField specifies the name of the field to be replaced.

szData specifies the data to be placed in the field.

iAddTo indicates if the data is to be appended to the existing data. A value of 1 will append the data. A value of 0 will overwrite the data.

Return Values

The GMW_DB_Replace function returns the following values:

GMW_DB_Replace Return Values

Return	Description
0	Error occurred
1	Field was successfully replaced

Unlocking a Record

GMW_DB_Unlock unlocks a record previously locked by a call to either GMW_DB_Append or GMW_DB_Replace.

Syntax

C/C++	long GMW_DB_Unlock(long pArea);
VB	Public Declare Function GMW_DB_Unlock Lib "gm6s32.dll" (ByVal lArea As Long) As Long

Parameter

The GMW_DB_Unlock function takes only pArea, which is the work area handle of the file opened by the GMW_DB_Open function.

Return Values

The GMW_DB_Unlock function returns the following values:

GMW_DB_Unlock Return Values

Return	Description
0	Error occurred
1	Success

Creating a Subset of Records

GMW_DB_Filter limits access to data in a GoldMine database by creating a subset of records based on expression criteria. If successfully called, all other functions (Top, Bottom, Skip, and so on) will respect the filter.

Syntax

C/C++	long GMW_DB_Filter(long pArea, char *szFilterExpr);
VB	Public Declare Function GMW_DB_Filter Lib "gm6s32.dll" (ByVal lArea As Long, ByVal strFilterExpr As String) As Long

Parameters

pArea is the work area handle of the file opened by the GMW_DB_Open function.

szFilterExpr is the valid Xbase expression. To remove the filter, send an empty string as the second parameter.

Return Values

The GMW_DB_Filter function returns the following values:

GMW_DB_Filter Return Values

Return	Description
0	Error occurred
1	Success

Limiting Search Scope

GMW_DB_Range activates the index in a table and sets a range of values to limit the scope of data that GoldMine will search. This function is faster than GMW_DB_Filter.

The Min and Max values must be formatted the same as the selected index tag's expression.

If successfully called, all other functions (Top, Bottom, Skip, etc.) will respect the range.

Syntax

C/C++	long GMW_DB_Range(long pArea, char *szMin, char *szMax, char *szTag);
VB	Public Declare Function GMW_DB_Range Lib "gm6s32.dll" (ByVal lArea As Long, ByVal strMin As String, ByVal strMax As String, ByVal strTag As String) As Long

Parameters

pArea is the work area handle of the file opened by the GMW_DB_Open function.

szMin specifies the minimum or lower value of the range.

szMax specifies maximum or upper value of the range.

szTag is the index tag name.

Return Values

The GMW_DB_Range function returns the following values:

GMW_DB_Range Return Values

Return	Description
0	Error occurred
1	Success

Performing a Sequential Search

GMW_DB_Search performs a sequential search on a file.

Syntax

C/C++	long GMW_DB_Search(long pArea, char *szExpr, char *szRecID);
VB	Public Declare Function GMW_DB_Search Lib "gm6s32.dll" (ByVal lArea As Long, ByVal strExpr As String, ByVal strRecID As String) As Long

Parameters

pArea is the work area handle of the file opened by the GMW_DB_Open function.

szExpr is the valid Xbase expression. For a record to be “found” this expression must result as TRUE.

szRecID is the buffer where the return value is stored. The return value will be a record number under Xbase or a RecID under SQL. You may pass NULL as the third parameter if you do not want the RecNo/RecID.

Return Values

The GMW_DB_Search function returns the following values:

GMW_DB_Search Return Values

Return	Description
0	No match found
>0	**Xbase:** RecNo of the matching record; **SQL:** RecID of the matching record

Moving to the First Record Match

GMW_DB_Seek positions to the first record matching the seek value.

Syntax

C/C++	long GMW_DB_Seek(long pArea, char * szParam);
VB	Public Declare Function GMW_DB_Seek Lib "gm6s32.dll" (ByVal lArea As Long, ByVal strParam As String) As Long

Parameters

pArea is the work area handle of the file opened by the GMW_DB_Open function.

szParam is the value you will seek. This value must match the format of the index expression for the currently active index.

Return Values

The GMW_DB_Seek function returns the following values:

GMW_DB_Seek Return Values

Return	**Description**
0	Error occurred
1	Exact match found. Cursor moved to record.
2	Exact match not found. Cursor placed at closest matching record.
3	EOF (end of file)
4	BOF (beginning of file)

Setting the Current Index Tag

GMW_DB_SetOrder sets the current index tag on the table.

Syntax

C/C++	long GMW_DB_SetOrder(long pArea, char *szTag);

VB	Public Declare Function GMW_DB_SetOrder Lib "gm6s32.dll" (ByVal lArea As Long, ByVal strTag As String) As Long

Parameters

pArea is the work area handle of the file opened by the GMW_DB_Open function. For a list of index names, see .

szTag is the name of the index tag to activate on the table.

Return Values

The GMW_DB_SetOrder function returns the following values:

GMW_DB_SetOrder Return Values

Return	Description
0	Error occurred
1	Index successfully activated

Positioning the Record Pointer

GMW_DB_Move positions the record pointer to a particular record in a data file.

Syntax

C/C++	long GMW_DB_Move(long pArea, char *szCommand, char *szParam);
VB	Public Declare Function GMW_DB_Move Lib "gm6s32.dll" (ByVal lArea As Long, ByVal strCommand As String, ByVal strParam As String) As Long

Parameters

pArea is the work area handle of the file opened by the GMW_DB_Open function.

szCommand is the command to execute. Each of these commands has an independent function equivalent that is the preferred method to use. This function remains as a legacy to its DDE counterpart.

szParam is the scope or value for the command.

GMW_DB_Move Commands and Function Equivalents

Command	Parameter	Function Equivalents
TOP	Not required	GMW_DB_Top
BOTTOM	Not required	GMW_DB_Bottom
SKIP	Number of records to skip	GMW_DB_Skip
GOTO	Record Number/RecID	GMW_DB_Goto

SEEK	Search key value	GMW_DB_Seek
SETORDER	Index Tag	GMW_DB_SetOrder

Return Values

The GMW_DB_Move function returns the following values:

GMW_DB_Move Return Values

Return	Description
0	Error occurred
1	Exact match found. Cursor moved to record or index-activated.
2	Exact match not found. Cursor placed at closes matching record.
3	Cursor at end-of-file (EOF)
4	Cursor at beginning-of-file (BOF)

Moving to a Specified Record

GMW_DB_Goto positions to a specific record in the table.

Syntax

C/C++	long GMW_DB_Goto(long pArea, char *szRecNo);
VB	Public Declare Function GMW_DB_Goto Lib "gm6s32.dll" (ByVal lArea As Long, ByVal strRecNo As String) As Long

Parameters

pArea is the work area handle of the file opened by the GMW_DB_Open function.

szRecNo specifies where the cursor should be placed, and is either the Record number for Xbase or the RecID for SQL

Return Values

The GMW_DB_Goto function returns the following values:

GMW_DB_Goto Return Values

Return	Description
0	Error occurred
1	Exact match found. Cursor moved to record or Index activated.

2	Exact match NOT found. Cursor placed at closest matching record.
3	Cursor at end-of-file (EOF)
4	Cursor at beginning-of-file (BOF)

Moving to the First Record

GMW_DB_Top positions to the first record in the table.

Syntax

C/C++	long GMW_DB_Top(long pArea);
VB	Public Declare Function GMW_DB_Top Lib "gm6s32.dll" (ByVal lArea As Long) As Long

Parameter

The GMW_DB_Top function takes only pArea, which is the work area handle of the file opened by the GMW_DB_Open function.

Return Values

The GMW_DB_Top function returns the following values:

GMW_DB_TopReturn Values

Return	Description
0	Error occurred
1	Cursor moved to top of file

Moving to the Previous or Following Record

GMW_DB_Skip positions to the previous or following record in the table.

Syntax

C/C++	long GMW_DB_Skip(long pArea, int nSkip);
VB	Public Declare Function GMW_DB_Skip Lib "gm6s32.dll" (ByVal lArea As Long, ByVal lSkip As Long) As Long

Parameters

pArea is the work area handle of the file opened by the GMW_DB_Open function.

nSkip specifies the number records to skip. This value can be positive to move forward in the table or negative to move backwards.

Return Values

The GMW_DB_Skip function returns the following values:

GMW_DB_Skip Return Values

Return	Description
0	Error occurred
1	Cursor successfully moved
3	Cursor at end-of-file (EOF)
4	Cursor at beginning-of-file (BOF)

Moving to the Last Record

GMW_DB_Bottom positions to the last record in the table.

Syntax

C/C++	long GMW_DB_Bottom(long pArea);
VB	Public Declare Function GMW_DB_Bottom Lib "gm6s32.dll" (ByVal lArea As Long) As Long

Parameter

The GMW_DB_Bottom function takes only pArea, which is the work area handle of the file opened by the GMW_DB_Open function.

Return Values

The GMW_DB_Bottom function returns the following values:

GMW_DB_Bottom Return Values

Return	**Description**
0	Error occurred
1	Cursor positioned on the last record in the table

Seeking a Record

GMW_DB_QuickSeek wraps several other database functions to provide a quick and easy way to seek a record in the database.

Syntax

C/C++	long GMW_DB_QuickSeek(char *szTableName, char *szIndex, char *szSeekValue, char *szRecID);
VB	Public Declare Function GMW_DB_QuickSeek Lib "gm6s32.dll" (ByVal strTableName As String, ByVal strIndex As String, ByVal strSeekValue As String, ByVal strRecID As String) As Long

Parameters

szTableName is the name of the table to be opened.

szIndex is the index to use for the table.

szSeekValue is the seek expression to use.

szRecID is returned by the function. This is the RecID of the record found.

Return Values

The GMW_DB_QuickSeek function returns the following values:

GMW_DB_QuickSeek Return Values

Return	Description
-2	Invalid Index
-1	Invalid table
0	Failure
1	Success

Reading a Field Value

GMW_DB_QuickRead wraps several other database functions to provide a quick and easy way to read a field value from a record in the database.

Syntax

C/C++	long GMW_DB_QuickRead(char *szTableName, char *szRecID, char *szField, char *szValue, int iLen);
VB	GMW_DB_QuickRead Lib "gm6s32.dll" (ByVal strTableName As String, ByVal strRecID As String, ByVal strField As StringByVal strValue As String, ByVal iLen As Long) As Long

Parameters

szTableName is the name of the table to be opened.

szRecID is the RecID of the record from which to read.

szField is the Field name to return.

szValue is the value returned by the function.

iLen is the length of the returned data.

Return Values

The GMW_DB_QuickRead function returns the following values:

GMW_DB_QuickRead Return Values

Return	Description
-4	Invalid Fieldname
-3	RecID not found
-2	Invalid RecID
-1	Invalid table
0	Failure
1	Success

Replacing a Field Value

GMW_DB_QuickReplace wraps several other database functions to provide a quick and easy way to replace a field value from a record in the database.

Syntax

C/C++	long GMW_DB_QuickReplace(char *szTableName, char *szRecID, char *szField, char *szValue, int iAddTo);
VB	GMW_DB_QuickReplace Lib "gm6s32.dll" (ByVal strTableName As String, ByVal strRecID As String, ByVal strField As String, ByVal strValue As String, ByVal iAddTo As Integer) As Long

Parameters

szTableName is the name of the table to be opened.

szRecID is the RecID of the record to be updated.

szField is the Field name to replace.

szValue is the value to store in the field.

iAddTo indicates if the value data is to be appended (1) or replaced (0=default).

Return Values

The GMW_DB_QuickReplace function returns the following values:

GMW_DB_QuickReplace Return Values

Return	Description
-4	Invalid Fieldname
-3	RecID not found
-2	Invalid RecID
-1	Invalid table
0	Failure
1	Success

Updating Sync Logs with GMXS32.DLL

The GoldMine GMXS32.DLL provides a method to update GoldMine synchronization logs whenever an external application updates GoldMine data.

`GMXS32.DLL` offers the following synchronization functions:

GMW_UpdateSyncLog: Updates the sync log file

GMW_ReadImpTLog: Imports a prepared TLog import file

GMW_NewRecID: Gets a new RecID

GMW_SyncStamp: Converts sync stamp to time and converts time back to sync stamp

Updating the Sync Log File

Syntax

C/C++	int GMW_UpdateSyncLog(char *szTable, char *szRecID, char *szField, char *szAction)
VB	GMW_UpdateSyncLog Lib "gm6s32.dll" (ByVal strTable As String, ByVal strRecID As String, ByVal strField As String, ByVal strAction As String) As Long

Parameters

szTable specifies the table name (such as “Contact1”) or the table ID.

szRecID specifies the RecID of the updated record: the correct RecID must be passed, and the RecID value must be exactly 15 characters long.

szField specifies the name of the field that has changed. This parameter is only relevant when the Action parameter is U. szField is ignored when Action is N or D.

szAction should be N when a new record has been appended, D when a record has been deleted, or U when a field in a record has been updated.

Return Values

The GMW_UpdateSyncLog function returns the following values:

GMW_UpdateSyncLog Return Values

Return	Description
0	Error
1	New TLog entry created
2	New TLog entry updated
4	Field TLog entry created
8	Field TLog entry updated
16	Deleted record TLog entry created
32	New TLog Entry removed

Example

```
char szTable[10] = "CONTACT1";
char szField[12] = "KEY2";
char szRecID[20] = "\0";
char szAction = 'U';
GMW_NewRecID(szRecID,"JON" ); GMW_UpdateSyncLog( szTable, szRecID, szField,
szAction );
```

Importing a Prepared TLog Import File

GMW_ReadImpTLog reads the status of a TLog import file, then deletes the import file when the process is completed.

Syntax

C/C++	int GMW_ReadImpTLog(char *szFile, int bDelWhenDone, char *szStatus)
VB	Public Declare Function GMW_ReadImpTLog Lib "gm6s32.dll" (ByVal strFile As String, ByVal lDelWhenDone As Long, ByVal strStatus As String) As Long

Parameters

szFile specifies the import file name—see below for the import file structure.

IDeleteWhenDone specifies to delete the import file when the process has completed.

SzStatus buffer used to monitor the status of the process. Optional, can be NULL. If passed, the szStatus buffer must be at least 10 characters long.

Return Values

The GMW_ReadImpTLog function returns the following values:

GMW_ReadImpTLog Return Values

Return	Description
0	Failure
> 0	Success, total number of imported TLog records

Notes

GMW_LoadAPI or GMW_LoadBDE must be called before calling GMW_ReadImpTLog for the first time. GMW_ReadImpTLog is executed in a thread, so multiple calls can be made. Your application can determine when the imported process completes by setting the iDeleteWhenDone parameter to 1, and noting when the import file is deleted. The TLog import must have the structure shown in the following table.

TLog Import Structure

Field Name	Type	Length
Table ID	char	10
RecID	char	15
Field ID	char	10
Action ID	char	1

Example

```
char szImpFile[80] = "d:\\GoldMine\\tlogimp.dbf";
char szStatus[20] = "\0";
int iDeleteWhenDone = 1;
int nTotRead = GMW_ReadImpTLog(szImpFile, iDeleteWhenDone, szStatus );
```

Getting a New Record ID

GMW_NewRecID returns a new RecID in the szRecIDBuf.

Syntax

C/C++	char* GMW_NewRecID(char *szRecIDBuf, char *szUser)
VB	Public Declare Function GMW_NewRecID Lib "gm6s32.dll" (ByVal strRecID As String, ByVal strUser As String) As GMWStr

Parameters

szRecID specifies the application allocated buffer to contain the new RecID. The buffer must be at least 16 characters long.

szUser specifies the GoldMine user name.

Return Value

pointer to szRecIDBuf

Notes

GMW_NewRecID returns a new RecID in the szRecIDBuf. GMW_NewRecID can be called without first calling GMW_LoadAPI or GMW_LoadBDE.

Example

```
char szRecID[20] = "\0";
char szUser[10] = "JON";
GMW_NewRecID( szRecID, szUser );
```

Converting the Sync Stamp

GMW_SyncStamp converts Sync Stamp to time format and back.

Syntax

C/C++	int GMW_SyncStamp(char *szStamp, char *szOutBuf)
VB	Public Declare Function GMW_SyncStamp Lib "gm6s32.dll" (ByVal strStamp As String, ByVal strOutBuf As String) As Long

Parameters

When the szStamp string parameter is exactly 17 characters long, formatted as Date:Time in form of CCYYMMDD:HH:MM:SS, the return string in szOutBuf is in TLog timestamp format, exactly seven characters long. When the szStamp parameter is seven characters long formatted as a TLog timestamp, the return string in szOutBuf is formatted as CCYYMMDD:HH:MM:SS.

Return Values

The GMW_SyncStamp function returns the following values:

GMW_SyncStamp Return Values

Return	Description
0	Failure
1	Success

Notes

An empty return string indicates an error.

Example

The following examples convert February 1, 1998, at 7:01pm to a TLog time stamp format, then back to a date and time format:

```
Char szOut[20] = "\0"
GMW_SyncStamp("19980201:19:01:30", szOut); // returns "+#G><N2"
GMW_SyncStamp("+#G><N2", szOut ); // returns "19980201:19:01:30"
```

Working with the XML API

Overview

Beginning in GoldMine version 6.7, the GoldMine API can be accessed using XML via the GMXMLAPI.DLL. The programmer may pass XML generated programmatically by concatenating strings or by using the Document Object Model (DOM). XML provides a simple and flexible medium for passing and receiving data from GoldMine's API.

A DOM Parser, such as MSXML or Xerces, should be utilized in constructing the XML documents for the GoldMine XML API. All GoldMine data needs to be XMLEncoded to avoid conflicts with XML entities (ie. < > ' &). A DOM Parser would handle this, in addition to creating well-formed XML. Finally, some of the XML documents returned will be too large to be handled by manually looping through the XML; whereas a parser would make accessing the returned data much more manageable.

The GMXMLAPI.DLL is used independently of the GMXS32.DLL. The XML API exposes all of the functionality present in the GMXS32, including the low-level data access functions. However, the power of implementing an integration with XML allows the use of the GoldMine API in any development environment that supports COM, including VB, VB.NET, C++, C#, and JAVA.

This chapter will discuss how to login to GoldMine with the XML API , how to call the business logic functions, and accessing the low level data functions. For specific information on the names of the business logic functions and acceptable data parameters and their return values, see .

Executing Your XML Document

Once the XML document has been created, pass it to the GoldMine XML API with the ExecuteCommand method. This is the only method exposed in the XML API. It accepts one parameter, xmlIn (the XML document prepared by the developer) and returns the resulting XML document detailing result and/or error codes.

Example

```
xmlout = GMAPI.ExecuteCommand(xmlIn)
```

Creating Your XML Document

The root XML element for the GoldMine XML API is defined as the following:

```
<GMAPI call="FunctionName">
<data name="Parameter1">Parameter Value</data>
<data name="Parameter2">Parameter Value 2</data>
</GMAPI>
```

Loading the API (GoldMine 7.0 or higher)

The first function to execute is loading the API with the desired parameters. Calling the LoadAPI function will also login the specified user into the API.

NOTE: The GoldMine XML API will always use a GoldMine seat for each user that is logged into it. The total number of users logged into GoldMine will be all workstation users and add-on applications combined.

To load the API and login the user, create the following XML:

```
<GMAPI call="LoadAPI">
<data name="User">kevin</data>
<data name="Password">mygmpass</data>
<data name="SysDir">c:\program files\goldmine\</data>_
<data name="GoldDir">c:\program files\goldmine\gmbase\</data> _
<data name="ComDir">c:\program files\goldmine\common\</data> _
<data name="SQLUser">sa</data>_
<data name="SQLPassword"></data>
</GMAPI>
```

Parameters

The LoadAPI function takes seven parameters.

User: Specifies the GoldMine user name (case insensitive).
You may set this parameter to the value of *DDE_LOGIN_CREDENTIALS* to use login credentials returned for the user logged into a running copy of GoldMine through DDE or COM.

Password: Specifies the user's password (case insensitive).
You may set this to the return string from the GetLoginCredentials DDE or COM command if the User parameter is set to *DDE_Login_Credentials*. The credential string is only valid for 30 seconds.

SysDir: Specifies the location of the LICENSE.BIN file (Version 7.0 or l ater).

GoldDir: Specifies the location of the CAL table or the database alias name to use as the main database.

NOTE: The database alias name must be appended with a colon (":").

ComDir: Specifies the location of the CONTACT1 table or the database alias name to use as the contact set database.

NOTE: The database alias name must be appended with a colon (":").

SQLUser: The login name for the SQL Server, if applicable.

SQLPassword: The password for the SQL Server, if applicable.

NOTE: The GMXS32.DLL required the call of GMW_SetSQLUserPass prior to calling GMW_LoadBDE in order to set the SQL username and password. This extra call is not used in the XML API.

The returned XML from LoadAPI will indicate if the call succeeded, and if so, a SessionID. This session ID is used to reference this particular user's API session. This is important in applications where multiple users are logged into the API simultaneiously. Even if the integration will only have one user logged in at a time, the Session ID must still be referenced in future calls to the XML API.

```
<GMAPI SessionID="1" call="LoadAPI">
<status code="1">API loaded successfully</status>
</GMAPI>
```

The status code will always give a description as to the cause of any generated errors. The possible return codes are as follows.

LoadAPI Return Values

Return	Description
1	API loaded successfully
0	API already loaded
-1	API failed to load
-2	Cannot find license file
-3	Cannot load license file
-4	Cannot validate the license file username/password
-5	Invalid GoldDir
-6	Invalid CommonDir
-7	Failed to allocate the needed TLS slot
-8	General Failure
-9	No access to specified contact set for this user

Loading BDE (GoldMine 6.7)

The first function that needs to be executed is loading the Borland Database Engine. Calling the function to load BDE will also login the specified user into the API.

NOTE: The GoldMine XML API will always use a GoldMine seat for each user that is logged into it. The total number of users logged into GoldMine will be all workstation users and add-on applications combined.

To load the Borland Database Engine, create the following XML:

```
<GMAPI call="LoadBDE">
<data name="User">kevin</data>
<data name="Password">mygmpass</data>
<data name="SysDir">c:\program files\goldmine\</data>_
<data name="GoldDir">c:\program files\goldmine\gmbase\</data> _
```

```
<data name="ComDir">c:\program files\goldmine\common\</data> _
<data name="SQLUser">sa</data>_
<data name="SQLPassword"></data>
</GMAPI>
```

Parameters

The LoadBDE function takes seven parameters.

User: Specifies the GoldMine user name (case insensitive).
You may set this parameter to the value of *DDE_LOGIN_CREDENTIALS* to use login credentials returned for the user logged into a running copy of GoldMine through DDE or COM.

Password: Specifies the user's password (case insensitive).
You may set this to the return string from the GetLoginCredentials DDE or COM command if the User parameter is set to *DDE_Login_Credentials*. The credential string is only valid for 30 seconds.

SysDir: Specifies the location of the LICENSE.DBF.

GoldDir: Specifies the location of CAL.DBF.

ComDir: Specifies the location of CONTACT1.DBF.

SQLUser: The login name for the SQL Server, if applicable.

SQLPassword: The password for the SQL Server, if applicable.

NOTE: The GMXS32.DLL required the call of GMW_SetSQLUserPass prior to calling GMW_LoadBDE in order to set the SQL username and password. This extra call is not used in the XML API.

The returned XML from LoadBDE will indicate if the call succeeded, and if so, a SessionID. This session ID is used to reference this particular user's API session. This is important in applications where multiple users are logged into the API simultaneiously. Even if the integration will only have one user logged in at a time, the Session ID must still be referenced in future calls to the XML API.

```
<GMAPI SessionID="1" call="LoadBDE">
<status code="1">BDE loaded successfully</status>
</GMAPI>
```

The status code will always give a description as to the cause of any generated errors. The possible return codes are as follows.

LoadBDE Return Values

Return	Description
1	BDE loaded successfully
0	BDE already loaded
-1	BDE failed to load
-2	Cannot find license file

-3	Cannot load license file
-4	Cannot validate the license file username/password
-5	Invalid GoldDir
-6	Invalid CommonDir
-7	Failed to allocate the needed TLS slot
-8	General Failure
-9	No access to specified contact set for this user

Logging in Subsequent Users

If an additional user needs to be logged into the XML API, call the Login method.

```
<GMAPI call="Login">
<data name="User">MASTER</data>
<data name="password">ACCESS</data>
<data name="ComDir">c:\program files\goldmine\common\</data> _
<data name="SQLUser">sa</data>
<data name="SQLPassword">mypassword</data>
</GMAPI>
```

Parameters

The Login function takes five parameters.

User: Specifies the GoldMine user name (case insensitive).
You may set this parameter to the value of *DDE_LOGIN_CREDENTIALS* to use login credentials returned for the user logged into a running copy of GoldMine through DDE or COM.

Password: Specifies the user's password (case insensitive).
You may set this to the return string from the GetLoginCredentials DDE or COM command if the User parameter is set to *DDE_Login_Credentials*. The credential string is only valid for 30 seconds.

ComDir: Specifies the location of CONTACT1.DBF or the database alias name to use as the contact set database.

NOTE: The database alias name must be appended with a colon (":")..

SQLUser: The login name for the SQL Server, if applicable.

SQLPassword: The password for the SQL Server, if applicable.

The Login function returns the following XML:

```
<GMAPI SessionID="2" call="Login">
<status code="1">Login Successful</status>
</GMAPI>
```

Login Return Values

Return	Description
1	Success
0	Failure
-1	User does not have permission to open the current contact set.

Logging Out

To log out a user when multiple users are logged in, use the Logout function. This function will free the license seat previously used by the Login function. Be sure to call this function for each session that has been opened.

Syntax

XML	<GMAPI call="Logout" SessionID="2"/>

Parameters

SessionID is the integer value returned by the Login function.

Return

The function will return a code attribute of “1” if the specified SessionID was valid. The returned XML will look like the following:

```
<GMAPI SessionID="2" call="Logout">
<status code="1">Logout succeeded for the supplied session.</status>
</GMAPI>
```

Unloading the API (GoldMine 7.0 or higher)

Before ending your GoldMine integration application, the API needs to be unloaded. The XML to unload the API is as follows:

```
<GMAPI call="UnloadAPI" SessionID="1"/>
```

The actual SessionID will be the value that was returned by the LoadAPI call.

Unloading BDE (GoldMine 6.7)

Before ending your GoldMine integration application, the Borland Database Engine needs to be unloaded. The XML to unload the BDE is as follows:

```
<GMAPI call="UnloadBDE" SessionID="1"/>
```

The actual SessionID will be the value that was returned by the LoadBDE call.

Accessing Data with Business Logic Functions

Reading and modifying GoldMine data with the business logic functions is the best-practice method for integrating with GoldMine. For the XML root element, the call will be any business logic function name, as described in Chapter 6, Business Logic Functions. Each data name will be the name portion of the defined name/value pairs, and the text for that node is the value portion of a name/value pair. For example, to create a contact using the GoldMine XML API, one would create an XML document like the following:

```
<GMAPI call=”WriteContact” SessionID=”1”>
<data name=”Contact”>Sam Jackson</data>
<data name=”Company”>Jackson Plumbing</data>
<data name=”Phone1”>(123)456-7890</data>
</GMAPI>
```

Accessing Nested Nodes of Data

Some business logic functions require or return nodes that contain nested nodes. For example, if you wish to add members to a contact group, the XML would look like the following:

```
<GMAPI call="AddContactGrpMembers" SessionID="1">
<data name="GroupNo">1234</data>
<data name="Members">
<data name="AccountNo">A3042474804 WB9!JCat</data>
<data name="Reference">A Reference Value</data>
</data>
<data name="Members">
<data name="AccountNo">A3082867459(LP:#JGab</data>
<data name="Reference">Another Reference</data>
</data>
<data name="Members">
<data name="AccountNo">A3060244052#3?(N3Ste</data>
<data name="Reference">The last Reference Value</data>
</data>
</GMAPI>
```

Each time there needs to be an additional node for the Members node, simply repeat the Members node with the required data. This applies to any business logic function that requires more than one data value for a node, or more than one nested node.

Business Logic Function Return Values

The business logic functions will return the same return codes as described in Chapter 6, Business Logic Functions. An example of the XML returned is as follows:

***Input XML*:**

```
<GMAPI call="WriteContact" SessionID="1">
```

```
<data name="Contact">Joe Smith</data>
<data name="Company">Joes Window Washing</data>
<data name="phone1">3106548963</data>
</GMAPI>
```

Returned XML:

```
<GMAPI SessionID="1" call="WriteContact">
<status code="1">Success</status>
<data name="Return">
<data name="AccountNo">A4100552319*T_S{3Del</data>
<data name="COMPANY">Joes Window Washing</data>
<data name="CONTACT">Joe Smith</data>
<data name="PHONE1">3106548963</data>
<data name="RecID">AP7Q62B&amp;amp;*AK=3\T</data>
</data>
</GMAPI>
```

Accessing Low-level Data Manipulation Functionality

The following sections describe additional functions in the GoldMine XML API that allow data reading and updating via low-level methods. Use of the following functions requires in-depth knowledge of the GoldMine data structures and business rules. They are useful for accessing and writing data that is not accessible via the high-level business logic functions.

Retrieving Data with DataStream

DataStream returns the data of ordered records from any GoldMine table using the most efficient method available. The caller can specify:

- Fields and expressions to return
- Range of records to return
- Optional filter to apply to the data set

DataStream SQL query capabilities are very fast on SQL databases.

The DataStream method allows for many useful applications. One such group of applications would merge HTML templates with the data returned by GoldMine DataStream to publish the contents of GoldMine data on the Internet. Web pages can be created to display GoldMine data requested by a visitor. Based on visitor selections, a company could dynamically present a variety of HTML pages, including dealer addresses in a particular city, financial numbers stored in Contact2, and even seating availability at upcoming conferences. With a fast Internet connection and a strong SQL server, the GoldMine client could respond simultaneously to dozens of requests.

Advantages of Using DataStream

GoldMine DataStream is absolutely the fastest way to read data from GoldMine tables. Used correctly, DataStream will return the data faster than most development environments would directly. DataStream offers the following advantages:

- *Efficiency*: DataStream issues a single, most efficient SQL query or Xbase seek to retrieve records from the back-end database to the local client. On SQL databases, requests of a few hundred records could be sent from the server to the client with a single network transaction, greatly minimizing network traffic.
- *Speed*: All fields and expressions are parsed initially by DS_Range and DS_Query, and then quickly evaluated against each record in DS_Fetch. Other DDE methods (and development environments) require that each field be parsed and evaluated each time its data is read. This makes a big difference when reading hundreds or thousands of records.
- *Simplicity*: Only three function calls are required to read all the data. Using traditional record-by-record querying would require one call for each field of each record (reading 10 fields from 50 records would require 500 function calls).
- *Results*: All the work to gather and format the data is done in C++, which is the fastest method. The caller needs only to parse the resulting packet string.

DataStream Record Selection

The following DataStream functions are listed in the order in which they must be called.

DS_Range: Opens a ranged cursor

DS_Query: Opens an SQL query cursor

DS_Fetch: Fetches records

DS_Close: Closes cursor

Either the DS_Range function or the DS_Query function must be called first to request the data. These functions return the integer handle which must be passed to the DS_Fetch and DS_Close functions.

You must use either DS_Range or DS_Query—you cannot use both. The DS_Range and DS_Query functions execute equally fast on SQL databases. DS_Range executes much faster on Xbase tables than does DS_Query.

DS_Range

Syntax

XML

```
<GMAPI call = “DS_Range” sessionid=”X”>
<data name = "Table">CONTACT1</data>
<data name = "Tag">Contacc</data>
<data name="TopLimit"> A3042474804 WB9!Jcat</data>
<data name ="BotLimit"> A4090244569#H4J*3Dav</data>
<data name="Fields">CONTACT;COMPANY;PHONE1</data>
<data name="Filter"/>

</GMAPI>
```

DS_Range returns a range of records based on an index.

Parameters

The following parameters are required:

Table specifies the table name (such as “Contact1”) or the table ID.

Tag designates the tag that corresponds to the index file.

TopLimit specifies the top limit of the range. (Must conform to the index expression.)

BotLimit (or *BottomLimit*) specifies the bottom limit of the range. (Must conform to the index expression.)

Fields specifies the requested fields and expression to return—see "DS_Range Field Selection" on the following page.

The following parameter is optional:

Filter designates an optional Xbase filter expression.

Return Values

The XML returned by DS_Range will look like the following:

```
<GMAPI SessionID="2" call="DS_Range">
<status code="1">1</status>
</GMAPI>
```

The text of the code attribute is used as the "Area" or "Handle" value for DS_Fetch.

The DS_Range function returns the following values:

GMW_DS_Range Return Values

Return	Description
0	Failure
1–20	Success (handle)

DS_Range Field Selection

The Fields parameter passed to DS_Range should consist of the field names and Xbase expressions to evaluate against each record in the data set. Each field must be terminated with a semicolon (;). Xbase expressions must be prefixed with an ampersand (&), and terminated with a semicolon. Be sure to XML encode this as the ampersand is an XML entitiy.

DS_Query

Syntax

XML	`<GMAPI call ="DS_Query" SessionID ="1">`
	`<data name = "SQL">select recid from contsupp</data>`
	`<data name="Filter">xBase expression filter</data>`
	`</GMAPI>`

This function is very fast on SQL databases.

Parameters

SQL query sends the query for evaluation on the server. The SQL query can join multiple tables and return any number of fields.

Optional parameter *Filter* specifies a Boolean Xbase filter expression to apply to the data set (even on SQL tables), similar to the DDE SETFILTER command.

Return Values

The DS_Query function returns the following values:

DS_QueryReturn Values

Return	Description
0	Failure
-1	Invalid Query/Timeout
1–20	Success (handle)

DS_Fetch

DS_Fetch returns a single packet string containing the requested data from all records processed by the current "fetch" command.

Syntax

XML	`<GMAPI call="DS_Fetch" SessionID="3">`
	`<data name="Area">`*Value returned from Query or Range*`</data>`
	`<data name="RecordCount">50</data>`
	`<data name="Raw">1</data>`
	`</GMAPI>`

Parameters

RecordCount (or RecCount) specifies the number of records to return.

Area must be the value returned from DS_Range() or DS_Query().

Optional Parameters

FldDmt (or *FieldDelimiter*) specifies the field delimiter (default: carriage return). Omit this data node completely to use the default value.

RowDmt (or *RowDelimiter*) specifies the record delimiter (default: line feed). Omit this data node completely to use the default value.

Raw indicates the format the data should be returned as. The default ("0") puts the data into XML format. Setting Raw to "1" returns the data stream in the old return packet format, as described below.

For details about the packet format, see .

The XML Return packet

DS_Fetch has an option in the GoldMine XML API to return the data in an XML format that is easier to process than the traditional datastream return packet.

Consider the following DS_Query XML call:

```
<GMAPI call="DS_Query" SessionID="1">
<data name="SQL">select contact, company, key1 from contact1 where
contact='Rafael Zimberoff'</data>
<data name="Filter"/>
</GMAPI>
```

Return

```
<GMAPI SessionID="1" call="DS_Query"><status code="1">1</status></GMAPI>
```

The DS_Fetch call to retrieve the requested data is:

```
<GMAPI call="DS_Fetch" SessionID="1">
<data name="Area">1</data>
<data name="Raw">0</data>
<data name="RecordCount">25</data>
</GMAPI>
```

The resulting XML datastream return packet is:

```
<GMAPI SessionID="1" call="DS_Fetch">
<status code="1">Success</status>
<data name="Return">
<data name="Header">
<data name="field">
<data name="Field_Name">CONTACT</data>
<data name="Field_Type">C</data>
<data name="Field_Length">40</data>
<data name="Field_Decimal">0</data>
</data>
<data name="field">
<data name="Field_Name">COMPANY</data>
<data name="Field_Type">C</data>
<data name="Field_Length">40</data>
<data name="Field_Decimal">0</data>
</data>
<data name="field">
<data name="Field_Name">KEY1</data>
<data name="Field_Type">C</data>
<data name="Field_Length">20</data>
<data name="Field_Decimal">0</data>
</data>
</data>
<data name="CountData">3000-0001</data>
<data name="Rows">
<data Name="Row">
<data name="CONTACT">Rafael Zimberoff</data>
<data name="COMPANY">Z-Firm LLC</data>
<data name="KEY1">Partner</data>
```

```
</data>
</data>
</data>
</GMAPI>
```

The Header node contains child nodes for each field included in the SQL query, describing the fields' properties. The CountData node's text corresponds with the old fetch return packet's header data:

The first digit can be 0, 3, or 4:

0 indicates that more records are available, which could be fetched with another DS_Fetch call

3 indicates the end-of-file (EOF)

4 indicates the beginning-of-file (BOF)

Number following the dash indicates the total number of data records contained in the packet.

The Rows node contains a child node for each data record returned by the query.

DS_Fetch Return Packet

DS_Fetch returns a single packet string containing the data from all requested records. The packet includes a header record, followed by one record for each record evaluated by "fetch." Within each record in the packet, the fields are separated by a field delimiter specified in DS_Fetch. By default, the field delimiter is the carriage return character (13 or 0x0D).

The records in the packet are separated by the record delimiter. By default, the record delimiter is the line feed character by default (10 or 0x0A).

These delimiters are convenient when the requested data does not contain notes from blob fields. You can omit FldDmt and RowDmt to use the default delimiters. When requesting notes, override the default delimiters by passing other delimiter values to DS_Fetch. For packets with notes, good delimiters are the ASCII characters 1 and 2.

The XML example above might return xml similar to:

```
<GMAPI SessionID="3" call="DS_Fetch">
<status code="1">3000-0003
A3053029581%`O6B3Sim
A4082371189*&gt;$&gt;B3Vin
A4090244569#H4J*3Dav
</status>
</GMAPI>
```

The packet header record consists of two sections:

First byte can be 0, 3, or 4:

0 indicates that more records are available, which could be fetched with another DS_Fetch call

3 indicates the end-of-file (EOF)

4 indicates the beginning-of-file (BOF)

Number following the dash indicates the total number of data records contained in the packet.

DS_Close

DS_Close must be called when the operation is complete. Unclosed data streams will leak memory and leave the database connections needlessly open. Passing an Area (or Handle) of 0 closes all open DataStream objects.

Syntax

XML	`<GMAPI call="DS_Close" SessionID="4">`
	`<data name="Area">1</data>`
	`</GMAPI>`

DS_Close returns the following XML:

```
<GMAPI SessionID="4" call="DS_Close">
<status code="1">Success</status>
</GMAPI>
```

Accessing Low-Level Data Using Work Areas

The GoldMine XML API provides a complete set of functions that allow low-level access to the database tables. Using these functions, you can:

- Open particular data files
- Seek the values of the fields in the records in the data files
- Append records to the tables
- Delete records
- Replace data in the records

Database applications that need varied access to GoldMine data typically use this suite of functions. To work successfully, these functions rely on a work area parameter. Using this parameter, you can open multiple data files concurrently and manipulate each file independently by referencing the file by work area. These functions also maintain synchronization information, which is stored in the TLogs.

The GoldMine XML API offers the low-level access functions that are listed in the following table.

GMXS32.DLL Low-Level Access Functions

Function Name	Description
Opening and Closing Databases	
☐ DB_Open	Opens one GoldMine data file for processing by another application
☐ DB_Close	Releases a previously OPENed file when processing is complete
☐ DB_IsSQL	Determines whether the table is SQL (1) or Xbase (0)
Creating and Deleting Records	
☐ DB_Append	Adds a new, empty record to a GoldMine data file

☐ DB_Delete	Deletes the current record in the specified work area.
Reading and Writing Data	
☐ DB_Read	Queries a data file for the value of a field
☐ DB_RecNo	Determines either current record number position (Xbase), or the record ID (SQL)
☐ DB_Replace	Changes the value in a particular field in one GoldMine data file
☐ DB_Unlock	Unlocks a record previously locked by a call to either GMW_DB_Append or GMW_DB_Replace
Limiting Scope of Data	
☐ DB_Filter	Limits access to data in a GoldMine database by creating a subset of records based on expression criteria
☐ DB_Range	Activates the index in a table, and sets a range of values to limit the scope of data that GoldMine will search
Searching for Data	
☐ DB_Search	Performs a sequential search on a file
☐ DB_Seek	Positions to the first record matching the seek value
☐ DB_SetOrder	Sets the current index tag on the table
Navigating the Database	
☐ DB_Move	Positions the record pointer to a particular record in a data file
☐ DB_Goto	Positions to a specific record in the table
☐ DB_Top	Positions to the first record in the table
☐ DB_Skip	Positions to the next or prior record in the table
☐ DB_Bottom	Positions to the last record in the table

GMXS32.DLL Low-Level Access Functions

Function Name	Description
DB_QuickSeek	Wraps several DLL functions to perform a Seek based on an index
DB_QuickRead	Wraps several DLL function to perform a Read
DB_QuickReplace	Wraps several DLL functions to perform a Replace

For details, see or ”.

Opening a Data File

DB_Open opens one GoldMine data file for processing by another application. Be sure to call DB_Close after completing all operations on the open table. Failing to do so will cause the UnloadAPI or UnloadBDE function to wait indefinitely for the resource to close.

Syntax

XML	`<GMAPI call="DB_Open" SessionID=”1”>`
	`<data name="Table">Contact1</data>`
	`</GMAPI>`

Parameter

The DB_Open function takes only Table(or File), which is the name of the table to be opened.

Return Values

The XML returned by DB_Open for a successful call will look like the following:

```
<GMAPI SessionID="2" call="DB_Open">
<status code="1">76007040</status>
</GMAPI>
```

The code attribute will be 1 on success and the text of the attribute is the workarea to be used for subsequent low-level calls. If the call is unsuccessful, the code will be 0 and the text will indicate an error.

DB_Open Code Attribute Values

Code	Text
0	Error occurred
1	Work area handle for table, for example 57919176

Closing a Data File

DB_Close releases a previously opened file when processing is complete. All previously opened files must be properly closed—failure to do so can result in database errors.

Syntax

XML	`<GMAPI call="DB_Close" SessionID="2">`
	`<data name="Area">76007040</data>`
	`</GMAPI>`

Parameters

The DB_Close function takes only Area, which is the work area handle of the file opened by the DB_Open function.

Return Values

DB_Close returns the following XML on success:

```
<GMAPI SessionID="2" call="DB_Close">
<status code="1">Success</status>
</GMAPI>
```

Checking for an SQL Table

DB_IsSQL is used to determine if the table is MSSQL (1) or Other (0).

Syntax

XML	`<GMAPI call="DB_IsSQL" SessionID="3">`
	`<data name="Area">76021592</data>`
	`</GMAPI>`

Parameter

The DB_IsSQL function takes only Area, which is the work area handle of the file opened by the DB_Open function.

Return Value

The DB_IsSQL function returns the following values:

```
<GMAPI SessionID="3" call="DB_IsSQL">
<status code="0">The open file is xBase.</status>
</GMAPI>
```

DB_IsSQL Code Attribute Values

Code	Description
0	The open file is Other
1	The open file is MSSQL

Adding a Record

DB_Append adds an empty record to a GoldMine data file.

Syntax

XML	`<GMAPI call="DB_Append" SessionID="3">`
	`<data name="Area">76021592</data>`
	`</GMAPI>`

Before using DB_Append, you must open a data file using the DB_Open function. After executing the DB_Append function, the record pointer is positioned at the new empty record, and the record is locked and ready to accept field replacements.

When a CONTACT1 record is appended, GoldMine automatically fills in the new record with the appropriate ACCOUNTNO and CREATEBY values. For all other records, you must replace the ACCOUNTNO field with the value from the CONTACT1 record with which the new record is to be linked. The GoldMine XML API will automatically fill in the value of the RECID field.

Parameters

Area is the work area handle of the file opened by the DB_Open function.

Return Value

Xbase: APPEND function returns the record number of the new record as the code attribute, or 0 if the file could not be locked. The text of the code attribute is also the record number in xBase, Record ID in SQL and FireBird.

```
<GMAPI SessionID="3" call="DB_Append">
<status code="64">64</status>
</GMAPI>
```

SQL: APPEND function returns the RECID of the new record in the text of the code attribute. The code will be 1 or 0 indicating success or failure.

```
<GMAPI SessionID="3" call="DB_Append">
<status code="1">9NDJRJN(EQ[)JW:</status>
</GMAPI>
```

Deleting the Current Record

DB_Delete deletes the current record in the specified work area and moves the record pointer to the next record.

Syntax

XML	`<GMAPI call="DB_Delete" SessionID="4">`
	`<data name="Area">73140736</data >`
	`</GMAPI>`

Parameter

The DB_Delete function takes only Area, which is the work area handle of the file opened by the DB_Open function.

Return Value

The DB_Delete function returns the following XML:

```
<GMAPI SessionID="4" call="DB_Delete">
<status code="1">Success</status>
</GMAPI>
```

DB_Delete Code Attribute Values

Code	Description

0	Error occurred
1	Record deleted

Reading a Field Value

DB_Read queries a data file for the value of a field.

Syntax

XML	`<GMAPI call="DB_Read" SessionID="5">`
	`<data name="Area">73154424</data>`
	`<data name="Field">Company</data>`
	`</GMAPI>`

Parameters

Area is the work area handle of the file opened by the GMW_DB_Open function.

Field is the name of the field to read within the table.

Return Value

The XML returned for DB_Read using the sample XML above is as follows:

```
<GMAPI SessionID="5" call="DB_Read">
<status code="1">GoldMine, Inc.</status>
</GMAPI>
```

DB_Range Code Attribute Values

Code	Description
0	Error occurred
1	Success

Checking the Current Record Number or Record ID

DB_RecNo is used to determine either current record number position (Xbase) or the record ID (SQL or FireBird).

Syntax

XML	`<GMAPI call="DB_RecNo" SessionID="7">`
	`<data name="Area">73166392</data>`
	`</GMAPI>`

Parameters

Area is the work area handle of the file opened by the DB_Open function.

Return Value

Xbase: Returns the current record number

SQL: Returns the current RecID

The returned XML will look like the following:

```
<GMAPI SessionID="7" call="DB_RecNo">
<status code="1">BDNHWD5#0PA5]WV</status>
</GMAPI>
```

Changing a Field Value

DB_Replace changes the value in a particular field in one GoldMine data file. After all replace operations on a single record are complete, the record must be unlocked using DB_Unlock.

Syntax

XML	`<GMAPI call="DB_Replace" SessionID="9">`
	`<data name="Area">73177576</data>`
	`<data name="Field">Contact</data>`
	`<data name="NewValue">XML Contact</data>`
	`<data name="Append">0</data>`
	`</GMAPI>`

Parameters

Area is the work area handle of the file opened by the DB_Open function.

Field specifies the name of the field to be replaced.

NewValue specifies the data to be placed in the field.

Append indicates if the data is to be appended to the existing data. A value of 1 will append the data. A value of 0 will overwrite the data.

Return Value

The DB_Replace function returns the following XML:

```
<GMAPI SessionID="9" call="DB_Replace">
<status code="1">Success</status>
</GMAPI>
```

DB_Replace Code Attribute Values

Code	Description
0	Error occurred
1	Field was successfully replaced

Unlocking a Record

DB_Unlock unlocks a record previously locked by a call to either DB_Append or DB_Replace.

Syntax

XML	`<GMAPI call="DB_Unlock" SessionID="3">`
	`<data name="Area">75885408</data>`
	`</GMAPI>`

Parameter

The DB_Unlock function takes only Area, which is the work area handle of the file opened by the DB_Open function.

Return Value

The DB_Unlock function returns the following XML:

```
<GMAPI SessionID="3" call="DB_Unlock">
<status code="1">Success</status>
</GMAPI>
```

DB_Unlock Code Attribute Values

Code	Description
0	Error occurred
1	Success

Creating a Subset of Records

DB_Filter limits access to data in a GoldMine database by creating a subset of records based on expression criteria. This function is similar to DB_Range. If successfully called, all other functions (Top, Bottom, Skip, and so on) will respect the filter.

Syntax

XML	**<GMAPI call="DB_Filter" SessionID="1"> <data name="Area">57919176</data> <data name="Filter">contact1->contact="Paul Redstone"</data></GMAPI>**

Note

The Filter value above is XML encoded. Passing the value contact1->contact=”Paul Redstone” through an XML Parser would handle the XML encoding automatically.

Parameters

Area is the work area handle of the file opened by the GMW_DB_Open function.

Filter (or FilterExpr, Expr, Expression) is the valid Xbase expression. To remove the filter, send an empty string as the second parameter.

Return Value

The DB_Filter function returns the following XML:

```
<GMAPI SessionID="1" call="DB_Filter">
<status code="1">Success</status>
</GMAPI>
```

DB_Filter Code Attribute Values

Code	Description
0	Failure
1	Success

Limiting Search Scope

DB_Range activates the index in a table and sets a range of values to limit the scope of data that GoldMine will search. This function is faster than DB_Filter.

The Min and Max values must be formatted the same as the selected index tag's expression.

If successfully called, all other functions (Top, Bottom, Skip, etc.) will respect the range.

Syntax

XML

```
<GMAPI call="DB_Range" SessionID="1">
<data name="Area">57917464</data>
<data name="Min">A3042474804 WB9!JCat </data>
<data name="Max">A4090244569#H4J*3Dav</data>
<data name="Tag">Contacc"</data>
</GMAPI>
```

Parameters

Area is the work area handle of the file opened by the GMW_DB_Open function.

Min specifies the minimum or lower value of the range.

Max specifies maximum or upper value of the range.

Tag is the index tag name.

Return Value

The DB_Range function returns the following XML:

```
<GMAPI SessionID="1" call="DB_Range">
<status code="1">Success</status>
</GMAPI>
```

DB_Range Code Attribute Values

Code	Description
0	Error occurred
1	Success

Performing a Sequential Search

DB_Search performs a sequential search on a file.

Syntax

XML

```
<GMAPI call="DB_Search" SessionID="1">
<data name="Area">60211128</data>
<data name="Expression">contact1-&gt;contact="David
Evans"</data>
</GMAPI>
```

Parameters

Area is the work area handle of the file opened by the GMW_DB_Open function.

Expr (or *Expression*) is the valid Xbase expression. For a record to be “found” this expression must result as TRUE. Be sure to XML encode this, since the “>” in an Xbase expression is an XML entity.

Return Value

The DB_Search function returns the following XML:

```
<GMAPI SessionID="1" call="DB_Search">
<status code="1">23</status>
</GMAPI>
```

The text of the code attribute will be the record number for dBase databases, and the RecID for SQL databases.

DB_Search Code Attribute Values

Return	Description
0	No match found
1	Success – the text of the attribute will be: **Xbase:** RecNo of the matching record; **SQL:** RecID of the matching record

Moving to the First Record Match

DB_Seek positions to the first record matching the seek value. DB_SetOrder must be called at some point prior to calling DB_Seek in order to set an index tag.

Syntax

XML	`<GMAPI call="DB_Seek" SessionID="1">`
	`<data name="Area">60211128</data>`
	`<data name="Expression">A3100554903(ZUW)3Dav</data>`
	`</GMAPI>`

Parameters

Area is the work area handle of the file opened by the GMW_DB_Open function.

Param is the value you will seek. This value must match the format of the index expression for the currently active index.

Return Value

The DB_Seek function returns the following XML:

```
<GMAPI SessionID="1" call="DB_Seek">
<status code="1">Success- Exact match found.</status>
</GMAPI>
```

DB_Seek Return Values

Return	Description
0	Error occurred
1	Exact match found. Cursor moved to record.
2	Exact match not found. Cursor placed at closest matching record.
3	EOF (end of file)
4	BOF (beginning of file)

Setting the Current Index Tag

DB_SetOrder sets the current index tag on the table.

Syntax

XML	`<GMAPI call="DB_SetOrder" SessionID="1">`
	`<data name="Area">60211128</data>`
	`<data name="Tag">CONTACC</data>`
	`</GMAPI>`

Parameters

Area is the work area handle of the file opened by the DB_Open function. Tag is the name of the index tag to activate on the table. For a list of index names, see .

Return Value

The DB_SetOrder function returns the following XML:

```
<GMAPI SessionID="1" call="DB_SetOrder">
<status code="1">Success</status>
</GMAPI>
```

DB_SetOrder Code Attribute Values

Code	Description
0	Error occurred
1	Index successfully activated

Positioning the Record Pointer

DB_Move positions the record pointer to a particular record in a data file.

Syntax

XML	`<GMAPI call="DB_Move" SessionID="1">`
	`<data name="Area">60211128</data>`
	`<data name="Command">SKIP</data>`
	`<data name="Parameter">2</data>`
	`</GMAPI>`

Parameters

Area is the work area handle of the file opened by the GMW_DB_Open function.

Command is the command to execute. Each of these commands has an independent function equivalent that is the preferred method to use. This function remains as a legacy to its DDE counterpart.

Parameter is the scope or value for the command.

DB_Move Commands and Function Equivalents

Command	Parameter	Function Equivalents
TOP	Not required	DB_Top
BOTTOM	Not required	DB_Bottom
SKIP	Number of records to skip	DB_Skip
GOTO	Record Number/RecID	DB_Goto
SEEK	Search key value	DB_Seek
SETORDER	Index Tag	DB_SetOrder

Return Value

The DB_Move function returns the following XML:

```
<GMAPI SessionID="1" call="DB_Move">
<status code="1">Exact match found. Cursor moved to record or index
activated.</status>
</GMAPI>
```

DB_Move Code Attribute Values

Code	Description
0	Error occurred
1	Exact match found. Cursor moved to record or index-activated.
2	Exact match not found. Cursor placed at closest matching record.
3	Cursor at end-of-file (EOF)
4	Cursor at beginning-of-file (BOF)

Moving to a Specified Record

DB_Goto positions to a specific record in the table.

Syntax

XML

```
<GMAPI call="DB_Goto" SessionID="1">
<data name="Area">60211128</data>
<data name="RecordNumber">9Z2RME8(X%(!3\T</data>
</GMAPI>
```

Parameters

Area is the work area handle of the file opened by the GMW_DB_Open function.

RecNo (or *RecordNumber*) specifies where the cursor should be placed, and is either the Record number for Xbase or the RecID for SQL. The RecID works for Xbase as well.

Return Value

The DB_Goto function returns the following XML:

```
<GMAPI SessionID="1" call="DB_Goto">
<status code="1">Exact match found. Cursor moved to record or index
activated.</status>
</GMAPI>
```

DB_Goto Code Attribute Values

Return	Description
0	Error occurred
1	Exact match found. Cursor moved to record or Index activated.
2	Exact match NOT found. Cursor placed at closest matching record.
3	Cursor at end-of-file (EOF)
4	Cursor at beginning-of-file (BOF)

Moving to the First Record

DB_Top positions to the first record in the table. This function should not be called with an SQL database.

Syntax

XML	`<GMAPI call="DB_Top" SessionID="1">`
	`<data name="Area">60211128</data>`
	`</GMAPI>`

Parameter

The DB_Top function takes only Area, which is the work area handle of the file opened by the DB_Open function.

Return Value

The DB_Top function returns the following XML:

```
<GMAPI SessionID="1" call="DB_Top">
<status code="1">Success</status>
</GMAPI>
```

DB_Top Code Attribute Values

Code	Description
0	Error occurred
1	Cursor moved to top of file

Moving to the Previous or Following Record

DB_Skip positions to the previous or following record in the table.

Syntax

XML	`<GMAPI call="DB_Skip" SessionID="1">`
	`<data name="Area">60211128</data>`
	`<data name="Skip">3</data>`
	`</GMAPI>`

Parameters

Area is the work area handle of the file opened by the DB_Open function.

Skip specifies the number records to skip. This value can be positive to move forward in the table or negative to move backwards.

Return Value

The DB_Skip function returns the following XML:

```
<GMAPI SessionID="1" call="DB_Skip">
<status code="1">Success</status>
</GMAPI>
```

DB_Skip Code Attribute Values

Return	Description
0	Error occurred
1	Cursor successfully moved
3	Cursor at end-of-file (EOF)
4	Cursor at beginning-of-file (BOF)

Moving to the Last Record

DB_Bottom positions to the last record in the table.

Syntax

XML	`<GMAPI call="DB_Bottom" SessionID="1">`
	`<data name="Area">60211128</data>`
	`</GMAPI>`

Parameter

The DB_Bottom function takes only Area, which is the work area handle of the file opened by the DB_Open function.

Return Value

The DB_Bottom function returns the following XML:

```
<GMAPI SessionID="1" call="DB_Bottom">
```

```
<status code="1">Success</status>
</GMAPI>
```

DB_Bottom Code Attribute Values

Code	Description
0	Error occurred
1	Cursor positioned on the last record in the table

Seeking a Record

DB_QuickSeek wraps several other database functions to provide a quick and easy way to seek a record in the database.

Syntax

XML	`<GMAPI call="DB_QuickSeek" SessionID="1">`
	`<data name="Table">Contact1</data>`
	`<data name="Index">CONTACC</data>`
	`<data name="SeekValue">A3100554903(ZUW)3Dav</data>`
	`</GMAPI>`

Parameters

Table is the name of the table to be opened.

Index is the index to use for the table.

SeekValue is the seek expression to use.

Return Value

The DB_QuickSeek function returns the following XML:

```
<GMAPI SessionID="1" call="DB_QuickSeek">
<status code="1">9Z2RME8(X%(!3\T</status>
</GMAPI>
```

DB_QuickSeek Code Attribute Values

Return	Description
-2	Invalid Index
-1	Invalid table
0	Failure
1	Success – The text will be the recid of the found record.

Reading a Field Value

DB_QuickRead wraps several other database functions to provide a quick and easy way to read a field value from a record in the database.

Syntax

XML

```
<GMAPI call="DB_QuickRead" SessionID="1">
<data name="Table">Contact1</data>
<data name="Recid">9Z2RME8(X%(!3\T</data>
<data name="Field">Contact</data>
</GMAPI>
```

Parameters

Table is the name of the table to be opened.

RecID (or *RecordID*) is the RecID of the record from which to read.

Field (or *FieldName*) is the Field name to return.

Return Value

The DB_QuickRead function returns the following XML:

DB_QuickRead Code Attribute Values

Return	Description
-4	Invalid Fieldname
-3	RecID not found
-2	Invalid RecID
-1	Invalid table
0	Failure
1	Success

Replacing a Field Value

DB_QuickReplace wraps several other database functions to provide a quick and easy way to replace a field value from a record in the database.

Syntax

XML	`<GMAPI call="DB_QuickReplace" SessionID="1">`
	`<data name="Table">Contact1</data>`
	`<data name="Recid">9Z2RME8(X%(!3\T</data>`
	`<data name="Field">Key3</data>`
	`<data name="Data">Updated by XML API</data>`
	`<data name="AddTo">0</data>`
	`</GMAPI>`

Parameters

Table is the name of the table to be opened.

RecID (or *RecordID*) is the RecID of the record to be updated.

Field (or *FieldName*) is the Field name to replace.

Value (or *Data*, *NewValue*) is the value to store in the field.

AddTo (or *Append*) indicates if the value data is to be appended (1) or replaced (0=default).

Return Value

The DB_QuickReplace function returns the following XML:

```
<GMAPI SessionID="1" call="DB_QuickReplace">
<status code="1">Success</status>
</GMAPI>
```

DB_QuickReplace Code Attribute Values

Return	Description
-4	Invalid Fieldname
-3	RecID not found
-2	Invalid RecID
-1	Invalid table
0	Failure
1	Success

Returning Calendar Data

The ReadSchedule call returns all calendar data for a given RecID.

Syntax

XML	<GMAPI call="ReadSchedule" SessionID="XXX"> <data name="RecID">BUAQI6O!* C8]WV</data> </GMAPI>

Return Value

The ReadSchedule call returns the following XML:

```
<GMAPI call="ReadSchedule" SessionID="XXX">
 <status code="1">Success</status>
 <data name="Return">
 <data name="ACCOUNTNO">A5040658567&amp; _:+]Mat</data>
 <data name="ACTVCODE"/>
 <data name="COLORCODE">0</data>
 <data name="CONTACT">Matthew W &amp; Kathleen Blacklock</data>
 <data name="DURATION"> 30</data>
 <data name="LINK">1</data>
 <data name="LOPRECID"> UAQI6O((X$]]WV</data>
 <data name="NOTIFY">0</data>
 <data name="ONDATE">20060530</data>
 <data name="ONTIME"> 7:00am </data>
 <data name="PRIVATE">0</data>
 <data name="RECID">BUAQI6O!* C8]WV</data>
 <data name="RECTYPE">C</data>
 <data name="REF"/>
 <data name="RSVP">0</data>
 <data name="UPDATERELATED">0</data>
 <data name="USERID">GUY</data>
 </data>
</GMAPI>
```

For Sales-type records, The ReadSchedule call returns more data:

```
<GMAPI call="ReadSchedule" SessionID="XXX">
 <status code="1">Success</status>
 <data name="Return">
  <data name="ACCOUNTNO">A5040658567&amp; _:+]Mat</data>
  <data name="ACTVCODE">AA </data>
  <data name="AMOUNT">1110</data>
  <data name="COLORCODE">0</data>
  <data name="CONTACT">Matthew W &amp; Kathleen Blacklock</data>
  <data name="DURATION"> 30</data>
  <data name="LINK">1</data>
  <data name="LOPRECID"> UAQR0L&amp;6K]O]WV</data>
  <data name="NOTIFY">0</data>
  <data name="ONDATE">20060530</data>
  <data name="ONTIME"/>
  <data name="POTNSALE">1110</data>
  <data name="PRIVATE">0</data>
```

```
  <data name="PROBSALE">30</data>
  <data name="RECID">BUAQR0L(?B&amp;+]WV</data>
  <data name="RECTYPE">S</data>
  <data name="REF">Johnny Apple Sauce! </data>
  <data name="RSVP">1</data>
  <data name="UNITSSALE">2</data>
  <data name="UPDATERELATED">0</data>
  <data name="USERID">GUY</data>
 </data>
</GMAPI>
```

Updating Sync Logs

The GoldMine XML API provides a method to update GoldMine synchronization logs whenever an external application updates GoldMine data.

The GoldMine XML API offers the following synchronization functions:

UpdateSyncLog: Updates the sync log file

ReadImpTLog: Imports a prepared TLog import file

NewRecID: Gets a new RecID

SyncStamp: Converts sync stamp to time and converts time back to sync stamp

Updating the Sync Log File

Syntax

XML

```
<GMAPI call="UpdateSyncLog" SessionID="1">
<data name="Table">Contact1</data>
<data name="RecID">9NDJRJN(EQ[)JW:</data>
<data name="Field">Key3</data>
<data name="Action">U</data>
</GMAPI>
```

Parameters

Table specifies the table name (such as “Contact1”) or the table ID.

RecID specifies the RecID of the updated record: the correct RecID must be passed, and the RecID value must be exactly 15 characters long.

Field specifies the name of the field that has changed. This parameter is only relevant when the Action parameter is U. Field is ignored when Action is N or D.

Action should be N when a new record has been appended, D when a record has been deleted, or U when a field in a record has been updated.

Return Value

The UpdateSyncLog function returns the following XML:

```
<GMAPI SessionID="1" call="UpdateSyncLog">
```

```
<status code="4">Field TLog entry created.</status>
</GMAPI>
```

UpdateSyncLog Code Attribute Values

Return	Description
0	Error
1	New TLog entry created
2	New TLog entry updated
4	Field TLog entry created
8	Field TLog entry updated
16	Deleted record TLog entry created
32	New TLog Entry removed

Importing a Prepared TLog Import File

ReadImpTLog reads the status of a TLog import file, then deletes the import file when the process is completed.

Syntax

XML

```
<GMAPI call="ReadImpTLog" SessionID="1">
<data name="File">c:\tlogs\mytlog.dbf</data>
<data name="Delete">1</data>
</GMAPI>
```

Parameters

File specifies the import file name—see below for the import file structure.

Delete specifies to delete the import file when the process has completed.

Return Value

ReadImpTLog function returns the following values in the code attribute:

ReadImpTLog Code Attribute Values

Code	Description
0	Failure
1	Success -- Text is total number of imported TLog records

Notes

LoadAPI or LoadBDE must be called before calling ReadImpTLog for the first time. Your application can determine when the imported process completes by setting the Delete parameter to 1, and noting when the import file is deleted. The TLog import must have the structure shown in the following table.

TLog Import Structure

Field Name	Type	Length
Table ID	char	10
RecID	char	15
Field ID	char	10
Action ID	char	1

Getting a New Record ID

NewRecID returns a new RecID in the text of the code attribute of the returned XML.

Syntax

XML

```
<GMAPI call="NewRecID" SessionID="1">
<data name="User">KEVIN</data>
</GMAPI>
```

Parameters

User specifies the GoldMine user name.

Return Value

```
<GMAPI SessionID="1" call="NewRecID">
<status code="1">AQN8HK0 I9&amp; =$R</status>
</GMAPI>
```

Notes

The resulting Recid is XML encoded because it contains an XML entity. Reading the text of the code attribute via an XML Parser would return the correctly XML unencoded RecID.

Converting the Sync Stamp

SyncStamp converts Sync Stamp to time format and back.

Syntax

XML

```
<GMAPI call="SyncStamp" SessionID="1">
<data name="Stamp">19980201:19:01:30</data>
</GMAPI>
```

Parameters

When the Stamp parameter is exactly 17 characters long, formatted as Date:Time in form of CCYYMMDD:HH:MM:SS, the return string in the code attribute's text is in TLog timestamp format, exactly seven characters long. When the Stamp parameter is seven characters long formatted as a TLog timestamp, the return string in the code attribute's text is formatted as CCYYMMDD:HH:MM:SS.

Return Value

The SyncStamp function returns the following example XML:

```
<GMAPI SessionID="1" call="SyncStamp">
<status code="1">5V1QM50</status>
</GMAPI>
```

SyncStamp Code Attribute Values

Code	Description
0	Failure
1	Success

Notes

An empty return string indicates an error.

Using MSXML to Handle GoldMine API XML

MSXML is just one DOM parser that can be used to format and parse the XML to pass to the GoldMine XML API. This section will give a brief tutorial of functions that can be used to handle the GoldMine XML document. It does not comprehensively document MSXML; please refer to Microsoft's Developer Network (MSDN) for complete MSXML documentation. Another parser that is available is Xerces.

Getting Started

The examples in this section will use functions and syntax from Microsoft XML 4.0 and Visual Basic 6.0. Include a reference to Microsoft XML, v. 4.0 in your development project. To create a document reference, use the following code:

```
Dim doc As DOMDocument40
Set doc = New DOMDocument40
```

The XML document is now ready to be composed.

Defining the Root Element

The root element for the GoldMine XML API is GMAPI. The code below sets this value:

```
Dim xmlIn As String
```

```
xmlIn = "<GMAPI/>"
Dim doc As DOMDocument40
Set doc = New DOMDocument40

doc.loadXML xmlIn

Dim elRoot As IXMLDOMElement
Set elRoot = doc.documentElement
```

Creating an IXMLDOMElement object and setting it to doc.documentElement provides a reference to the root element of the document. This allows for easy updating to that element later on.

Setting Attributes

The attributes of an element define a specific setting or provide additional information to an element. Attributes appear in an element's start tag and are in a name/value pair format. The GoldMine XML API typically expects two attributes for the root element: call and sessionid.

To set an attribute, use the SetAttribute method in the documentElement object. The following code assumes the elRoot object defined above.

```
elRoot.setAttribute "call", "DB_Open"
elRoot.setAttribute "SessionID", sSessionID
```

Referencing an Attribute

The call attribute for the GMAPI root element will likely need to be changed many times in the course of your application. A reference to this attribute can be obtained by calling the following code:

```
Dim att As IXMLDOMAttribute
Set att = elRoot.selectSingleNode("@call")
```

Now the GoldMine XML API call can be changed easily.

```
att.Text = "DB_Append"
```

IMPORTANT: Be sure to set all references to Nothing (or Null) before exiting your application!

```
Set elRoot = Nothing
Set doc = Nothing
Set att = Nothing
```

Creating Child Elements

To specify parameters of the GoldMine XML API function calls, a "data" element needs to be created for each parameter. Each data element has one attribute titled "name". The value of the parameter is stored as the text value of the attribute. Following is a Visual Basic example showing a subroutine that sets a parameter for the GoldMine XML API:

```
Public Sub SetParameter(doc As DOMDocument40, root As IXMLDOMElement,
 sParamName As String, ByVal sValue As String)
```

```
Dim tempEL As IXMLDOMElement

'Create the element and assign to a reference
Set tempEL = doc.createElement("data")

'Set the attribute with the sParamName value being the name of the
'parameter
tempEL.setAttribute "name", sParamName

'Specify the value of the parameter
tempEL.Text = sValue

'Append the child element to the root
root.appendChild tempEL
Set tempEL = Nothing

End Sub
```

The above subroutine can now be called to set many parameters for a function. The example below assumes the document, root element and attribute objects created in the previous section.

```
att.Text = "DB_Replace"

SetParameter doc, elRoot, "Field", "Contact"
SetParameter doc, elRoot, "NewValue", "XML Contact"
SetParameter doc, elRoot, "Append", "0"
```

Executing the XML Document

The GoldMine XML API exposes a single method to execute the XML document: ExecuteCommand. The following subroutine wraps the calls necessary to execute the API's XML:

```
Public Sub ExecuteCommand(doc As DOMDocument40)
Dim strOut As String

Dim GMAPI As GMXMLAPI.GoldMineData

Set GMAPI = New GMXMLAPI.GoldMineData
strOut = GMAPI.ExecuteCommand(doc.xml)

'xmlout is a global string variable. This can be changed to be 'returned by
the function call.
xmlout = strOut
Set GMAPI = Nothing

End Sub
```

Reading the Results

The GoldMine XML API returns the results of the function calls by adding an element called status with an attribute called “code”. In addition, data returned by the call, such as contact information, is returned as child elements.

Reading the Code Attribute

After executing an XML API command, the resulting XML document contains a status element with a code attribute. The value of this attribute represents the return value of the function executed. The text value of the code attribute is a description of the return value, typically providing a meaningful explanation of an error code. The following subroutine returns the code as the return value and the textual description as an optional output parameter:

```
Public Function GetReturnVal(Optional sDescription As String) As Integer
Dim DomDoc As DOMDocument40
Set DomDoc = New DOMDocument40

‘xmlout is a global variable that contains the returned XML from
 ‘the ExecuteCommand subroutine defined in the above section
DomDoc.loadXML xmlout

Dim root As IXMLDOMElement
Set root = DomDoc.documentElement
If root.Attributes.length > 0 Then
Dim status As IXMLDOMNode
Set status = root.childNodes(0)
If status.Attributes(0).baseName = "code" Then
sDescription = status.Text
GetReturnVal = status.Attributes(0).Text
End If
End If

Set DomDoc = Nothing

Set root = Nothing
Set status = Nothing

End Function
```

Reading the Returned Data

The GoldMine XML API returns an element titled “Return” containing the data elements returned by the executed command. The best way to access the individual elements using MSXML is to call selectsingleNode and specify an XPath expression to designate the desired element. SelectsingleNode returns a reference to the element requested. To access a further-nested element, call selectsingleNode again from the originally returned element. The following code loads an XML document returned from executing the ReadRecord command. It then obtains a reference to the “Return” element, followed by requesting the “CONTACT” element from the “Return” element.

```
Dim elReturnData As IXMLDOMElement
Dim elFieldValue As IXMLDOMElement
Dim docReturned As DOMDocument40
Dim elRootReturned As IXMLDOMElement
```

```
Set docReturned = New DOMDocument40

docReturned.loadXML xmlReturned
Set elRootReturned = docReturned.documentElement

Set elReturnData = elRootReturned.selectSingleNode("data[@name='Return']")
If Not elReturnData Is Nothing Then
Set elFieldValue = elReturnData.selectSingleNode("data[@name='CONTACT']")
If Not elFieldValue Is Nothing Then _
txtContactName = elFieldValue.Text
End If

Set elReturnData = Nothing
Set elFieldValue = Nothing
Set elRootReturned = Nothing
Set docReturned = Nothing
```

The XPath expression is case sensitive. Typically, all field name elements will be in ALL CAPS. Other element names may not be formatted in that manner. The case format of the element name can be checked by inspecting the returned XML during the design phase of your application.

Accessing the Current GoldMine Instance with COM

Overview

With the release of GoldMine 6.7, GoldMine acts as a COM Server. This new functionality enables an application to interact with GoldMine without using DDE or loading a dll. In addition, integrating your application with GoldMine using the COM Server ability does not require a separate instance of Borland Database Engine (BDE) to be loaded. Furthermore, utilization of the COM server in GoldMine allows the integrating application to control GoldMine's user interface to a much greater extent than the legacy DDE server allowed.

NOTE: As of GoldMine Version 7.0, the Borland Database Engine is no longer used. References to BDE in this chapter apply to integrations developed in GoldMine Version 6.7.

All COM server class methods are executed via XML. For information on using Microsoft XML for creating XML documents to use with the GoldMine COM Server, please see .

There are 3 classes exposed by the COM server:

1. *GoldMine.GoldMineData* – This class has methods that are exactly as in the GoldMine XML API described in Chapter 4, Working with the XML API. However, this class does not contain any functions for loading BDE or logging in, as they are unnecessary with a running instance of GoldMine. Using the GoldMine.GoldMineData class of the COM Server will alleviate the SharedMemLocation BDE setting issues with loading a second BDE instance.

NOTE: Since these commands are an exact duplicate to the GoldMine XML API commands, they will not be documented in this chapter. For information on using the commands accepted in this class, please see .

2. *GoldMine.UI* – This class has methods and events that replace all current DDE functionality and to control the GoldMine user interface.
3. *GoldMine.RecObj* – This class has events for notifying client applications of Record object changes.

Getting Started

To access the GoldMine COM Server, add a reference to the GoldMine 6.7 Type Library to your project. Objects for each of the classes can now be created.

```
Dim WithEvents GMUI As GoldMine.UI
Dim WithEvents RcOb As GoldMine.RecObj
Dim GMData As GoldMine.GoldMineData
```

In addition, your application needs to be COM Exception aware.

For instance if a login fails, then a COM Exception of type AccessDenied is passed to your application.

Executing Commands

The GoldMine.UI and GoldMine.GoldMineData classes only have one exposed method:

```
ExecuteCommand([in]BSTR xmlIn, [out, retval] BSTR* xmlOut)
```

To use this method, build your XML document using a DOM parser, such as MSXML, then pass the resulting document to the ExecuteCommand method.

```
strOut = GMUI.ExecuteCommand(txtXMLIn.Text)
```

- If your application is developed in VB, C#, VB.NET, or Delphi the call will have the same format as above.

  StringVar = GMUI.ExecuteCommand(xmlIN)

- If your application is developed in C++, or another lower-level programming language, the call will have the format of:

  ExecuteCommand(xmlIn, xmlOut)

Logging In to GoldMine

Using the GoldMine COM Server requires that GoldMine is running on the computer the client application is also running on. If GoldMine is not running, it will be launched the first time a call is made to the GoldMine COM Server. However, this will only bring GoldMine to the login screen. The GoldMine.UI and GoldMine.GoldMineData classes both have a command to handle this, Login. Following is example code for calling the Login command:

```
GMObj.ExecuteCommand("<GMAPI call=""Login""><data
name=""User"">MASTER</data><data name=""Pass"">ACCESS</data></GMAPI>")
```

- If GoldMine is already running, the COM server will return:

```
<GMAPI call="Login">
<status code="-31703">The call passed was not recognised as valid.</status>
</GMAPI>
```

- If the Login attempt was successful, the COM server will return:

```
<GMAPI call="Login">
<status code="1">Succeeded.</status>
</GMAPI>
```

- If invalid login information is passed, a COM Exception of type AccessDenied is returned to the client application.

GoldMine.UI Class

The UI class of the GoldMine COM Server provides identical functionality to the legacy DDE Server. If you are familiar with using the DDE commands, porting to the COM Server will be natural. There is additional functionality in the COM Server that allows control of the GoldMine user-interface with commands such as launching menu items, being notified when a window is being launched, and manipulating controls.

Accessing Data Files

GoldMine.UI provides a complete set of commands that allow low-level access to the data files. These functions allow you to:

- Open particular data files,
- Query the values of the fields in the records in the data files,
- Add records to the files, and
- Replace data in the records.

This suite of functions is usually used for database applications that need varied access to GoldMine data.

Adding an Empty Record

Syntax	`<GMAPI call="Append">`
	`<data name="Area">1</data>`
	`</GMAPI>`

The Append function is used to add an empty record to a GoldMine data file. Before using Append, you must open a data file using the Open function. After executing the Append function, the record pointer is positioned at the new empty record, and the record is locked and ready to accept field replacements.

When a CONTACT1 record is appended, GoldMine automatically propagates the new record with the appropriate ACCOUNTNO and CREATEBY values. For all other records, you must replace the ACCOUNTNO field with the value from the CONTACT1 record with which the new record is to be linked. For records that require remote synchronization initialization, GoldMine will automatically propagate the value of the RECID field when these records are appended.

Parameters

The Append function accepts one parameter, the work area handle of the file to Append. The work area handle is returned by the Open file when the file is opened.

Return Value

Xbase: The Append function returns the record number of the new record, or 0 if the file could not be locked.

SQL: The Append function returns the record ID.

Returned XML

```
<GMAPI call="Append">
<status code="1">72</status>
```

```
</GMAPI>
```

Closing an Opened File

Syntax	<GMAPI call=”Close”> <data name=”Area”> 1</data> </GMAPI>

```
<GMAPI call=”Close”>
<data name=”Area”> 1</data>
</GMAPI>
```

The Close function is used to release a previously OPENed file when processing is complete. When access is complete, a file must be CLOSEd to release memory used by GoldMine to maintain database work areas.

Parameters

The Close function accepts one parameter, Area—the work area handle of the file to close. The Open file returns the work area handle when the file is opened.

Return Value

The Close value returns 1 if the function was able to successfully close the work area, 0 if an invalid work area handle was passed.

Returned XML

```
<GMAPI call="Close"><status code="1">Success</status></GMAPI>
```

Deleting the Current Record

Syntax	<GMAPI call=”Delete”> <data name=”Area”>1</data> </GMAPI>

The Delete function deletes the current record in the specified work area. The record pointer is not advanced to the next record.

Parameters

The Delete function takes one parameter, Area—the work area value obtained from the Open function.

Returned XML

```
<GMAPI call="Delete">
<status code="1">Success</status>
</GMAPI>
```

Creating a Subset of Records

Syntax	<GMAPI call=”Filter> <data name=”Area”>1</data> <data name=”Expression”>Xbase Expression</data> </GMAPI>

The Filter function limits access to data in a GoldMine database by creating a subset of records based on expression criteria.

Parameters

The Filter function takes two parameters.

Area: the work area handle of the file that you want to read. The Open function provides this value when the data file is opened.

Expression: a valid Xbase expression. Referencing a table and field in an Xbase expression requires the use of the ">" character. Since this is an XML entity, be sure to build this XML document through a DOM parser to XML encode the elements. See for more information.

To remove the filter from the database, use a Filter function with an empty string, such as:

```
<GMAPI call="Filter">
<data name="Area">1</data>
<data name="Expression"/>
</GMAPI>
```

Checking for an Xbase or SQL Table

Syntax	`<GMAPI call="IsSQL">`
	`<data name="Area">1</data>`
	`</GMAPI>`

The IsSQL function returns the table type (Xbase or SQL) that is open in a work area. Using this command, you can determine the most appropriate method to retrieve information when working with DataStream. For example, when your routine starts, you can open Contact1 and Cal, issue an IsSQL command to determine the GoldDir and CommonDir database types, and then close both work areas. You can then send the appropriate DataStream calls.

Parameters

The IsSQL function takes work area as the only parameter, Area.

Return Value

IsSQL returns 1 for an SQL database table, or 0 for an Xbase file.

Returned XML

```
<GMAPI call="IsSql">
<status code="0">The open file is xBase.</status>
</GMAPI>
```

Moving to a Specified Record

Syntax	`<GMAPI call="Move">`
	`<data name="Area"> 87494472</data>`
	`<data name="Command">COMMAND</data>`
	`<data name="Parameter">PARAMETER</data>`
	`</GMAPI>`

The Move function will position the record pointer to a particular record in a data file. Before using Move, you must open a data file using the Open function.

Parameters

The Move function requires either two or three parameters.

Area: the work area handle of the file whose record pointer you want to position. The Open function provides this value when the data file is opened.

Command: the name of the Move subfunction that you want to perform.

Parameter: Depending on the subfunction, a third parameter can be required.

The following table lists the Move subfunctions and the requirements for the third parameter:

Valid Move Subfunctions

Subfunction	Description	3rd Parameter
TOP	Move to first logical record	Not required
BOTTOM	Move to last logical record	Not required
SKIP	Skip records	Optional, records to skip
GOTO	Go to a specific record	Record number (Xbase), Record ID (SQL)
SEEK	Seek a specific record by key	Search key value
SETORDER	Select an index	Index name

Top	Positions the record pointer at the first logical record according to the current index order. For example, if the data file open in the selected work area is CONTACT1.DBF, and the index order is set to *Company*, a call to TOP will result in the record pointer being positioned at a record with a company name, such as AAA Cleaners.
Bottom	Positions the record pointer at the last logical record according to the current index order. For example, if the data file open in the selected work area is CONTACT1.DBF, and the index order is set to *Company*, a call to BOTTOM will result in the record pointer being positioned at a record with a company name, such as Z-best Bakery.
Skip	Moves the record pointer record by record. If SKIP is called without the third parameter, it will move the record pointer to the next logical record according to the current index order. If SKIP is called with a string numeric as the third parameter, the record pointer will be moved forward by the indicated number if the value is positive, or backward if the value is negative. Negative numbers must be passed in quotation marks, for example “-1”.
Goto	Positions the record pointer at the record number (Xbase) or record ID (SQL) specified by a string numeric passed as the third parameter.

Seek	Attempts to locate a record in the data file with an index key that matches the string passed as the third parameter. Partial key searches are allowed; GoldMine will position the record pointer at the record with the key that most closely matches the passed value.
Setorder	Selects an active index for ordering and SEEKing the data file. See for the appropriate values and collating sequence for each data file index.

TIP: If an invalid index is selected for the data file, none of the MOVE subfunctions will operate properly.

Return Value

The Move function can return several values.

Move Return Values

Return	Description
0	Error occurred
1	Record pointer successfully moved, or index selected
2	Exact match not found, pointer positioned at closest match
3	Record pointer positioned at end-of-file (EOF)
4	Record pointer positioned at beginning-of-file (BOF)

An error can be returned under any of the following conditions:

- Invalid work area handle is passed to the function.
- Invalid subfunction is passed.
- Out-of-range record number is passed.
- Nonnumeric value is passed as a third parameter when a numeric value is expected.

Returned XML

```
<GMAPI call="MOVE">
<status code="1">1</status>
</GMAPI>
```

Opening a Data File

Syntax	<GMAPI call=”Open”> <data name=”Filename”>CONTACT1</data> </GMAPI>

The Open function is used to open a GoldMine data file for processing by another application. This function must be called before calling any GoldMine.UI data functions that work with an individual data file. It is not necessary to use this function when calling the RecordObj function or user-interface control functions.

Parameters

The Open function takes one parameter, Filename. The following values are valid for this parameter:

Open Valid Parameters

File	Description
CAL	Calendar activities file
CONTACT1	Primary contact information file
CONTACT2	Primary contact information file
CONTGRPS	Groups file
CONTHIST	History records file
CONTSUPP	Supplementary records file
INFOMINE	InfoCenter file
LOOKUP	Lookup file
MAILBOX	E-mail Center mailbox file
OPMGR	Opportunity Manager file
PERPHONE	Personal Rolodex file
RESOURCE	Resources file
SPFILES	Contact files directory

Return Value

The Open function returns an integer value representing the handle to the file's work area. This value is required for all subsequent access to the file. If the file could not be opened, or an invalid parameter is passed, the function will return 0.

Returned XML

```
<GMAPI call="Open"><status code="1">87732928</status></GMAPI>
```

Limiting GoldMine Search Range

Syntax

```
<GMAPI call="Range"> <data name="Area">87732928</data> <data
name="Min">Mark Durrant</data> <data name="Max">Paul
Redstone</data> <data name="Tag">CONTNAME</data></GMAPI>
```

The Range function activates the index in a table and sets a range of values to limit the scope of data that GoldMine will search.

Parameters

The Range function requires four parameters.

Area: the work area handle of the file that you want to read. The Open function provides this value when the data file is opened.

Min: the minimum value of the range.

Max: the maximum value of the range.

Tag: the tag that corresponds to the index file. For details about tags, see .

Returned XML

```
<GMAPI call="Range">
<status code="1">Success</status>
</GMAPI>
```

Syntax	`<GMAPI call="Query">`
	`<data name="Area">87732928</data>`
	`<data name="SQL">select recid from contact1 where state=”MI”</data>`
	`</GMAPI>`

The Query function limits the set of records that can be accessed to the result set from the specified SQL query. After calling the Query command, issue a MOVE command to move the record pointer into the result set from the Query (by calling TOP for example).

Parameters

Area: the area value returned by the Open command.

SQL: the SQL query to send to the server.

Returned XML

```
<GMAPI call="Query"><status code="1">Success</status></GMAPI>
```

Reading a Field Value

Syntax	`<GMAPI call="Read">`
	`<data name="Area">87624560</data>`
	`<data name="Field">Key1</data>`
	`</GMAPI>`

The Read function is used to query a data file for the value of a field. Before using Read, you must open a data file using the Open function. In addition, you will probably want to position the record pointer to the record you want to query by using the Move function.

Parameters

The Read function requires two parameters.

Area: The first parameter is the work area handle of the file that you want to read. The Open function provides this value when the data file is opened.

Field: The second parameter is the name of the field in the data file whose value you want to query. You will normally pass only a single field name, such as CONTACT as the second parameter. However, if you pass a field expression, such as "*COMPANY + CONTACT*" GoldMine will attempt to evaluate the expression and return the value of the expression.

Return Value

The Read function returns a character string containing the value in the specified field, or the value of the specified expression. An invalid work area handle, an invalid field being passed, or an expression that GoldMine could not evaluate can cause errors.

Returned XML

```
<GMAPI call="Read">
<status code="1">Client Prospect</status>
</GMAPI>
```

Checking the Current Record Number or Record ID

Syntax	`<GMAPI call="Recno">`
	`<data name="Area">87624560</data>`
	`</GMAPI>`

Xbase: RecNo function is used to determine current record number position.

SQL: RecNo function is used to determine the record ID.

Parameters

The RecNo function accepts one parameter, Area—the work area handle of the file. The Open function returns the workarea.

Return Value

The RecNo function returns the current record number position, 0 if an invalid work area handle was passed.

Returned XML

```
<GMAPI call="Recno">
<status code="1">21</status>
</GMAPI>
```

Changing a Field Value

Syntax

```
<GMAPI call="Replace">
<data name="Area">87637440</data>
<data name="Field">contact</data>
<data name="NewValue">Reuben Corazza</data>
<data name="Append">0</data>
</GMAPI>
```

The Replace function is used to change the value in a particular field in one GoldMine data file. Before using Replace, you must open a data file using the Open function. In addition, you will probably want to position the record pointer to the record you want to change either by using the Move function, or by adding a new record with the Append function.

After executing the Replace function, GoldMine will update the specified field with the new value, and update the appropriate remote synchronization data structures to indicate that the field was changed.

In a network environment, GoldMine automatically locks the record before performing the replacement. The record is not automatically unlocked, allowing for fast multiple field replacements. The record is automatically unlocked when a Close, Move, or Unlock command is issued on the work area.

Parameters

The Replace function requires three parameters and has an optional fourth parameter.

Area: The first parameter is the work area handle of the file in which you want to perform the replacement. The Open function provides this value when the data file is opened.

Field: The second parameter is the name of the field to be replaced. See for information on the name of fields in each GoldMine data files. If you attempt to replace a field that does not exist in the file open in the specified work area, the Replace function will fail.

NewValue: The third parameter is the value to replace. The replace value must be a string value. If the replacement field is a date or numeric field, GoldMine will convert the string data to the appropriate data type prior to performing the replacement.

Append: The fourth parameter will add data instead of replacing data. Using this parameter, you can insert large amount of text into a notes field. To append instead of replace incoming data from the third parameter, pass 1 as the fourth parameter. You can set up a loop to feed notes in 256-byte segments to override the 256-byte limit for inbound DDE requests.

Return Value

If the file was replaced, the Replace function returns 1.

```
<GMAPI call="Replace"><status code="1">Success</status></GMAPI>
```

If the field could not be replaced, 0 is returned. The failure can be caused under any of the following conditions:

- Invalid parameter, such as an invalid work area handle.
- Invalid field name.
- Record already locked by another user.

Performing a Sequential Search

Syntax

```
<GMAPI call="search">
<data name="area">87675752</data>
<data name="expression">contact="Paul Redstone"</data>
</GMAPI>
```

The Search function is used to perform a sequential search on a file. Unlike Move, Search scans the table, one record at a time, looking for a record that satisfies the search condition. The search condition can be any Xbase expression that GoldMine understands, but is usually an expression that tests the value of one or more fields in the file. When a match is found, the record pointer is located at the matching record.

Search starts with the record that immediately follows the current record (the next logical record according to the selected index order) and continues until a match is found or the end of file is encountered. Because of this, Search can be called repeatedly to return a list of records that satisfy the search condition.

Parameters

The Search function takes three parameters.

Area: the work area handle of the file you want to search. The Open function provides this value when the data file is opened.

Expression: the search expression, such as “CITY=‘Los Angeles’”

Return Value

The Search function can return several values.

Search Return Values

Return	Description
0	Error occurred or match could not be found
>0	Match found; return value indicated current physical record number (Xbase) or record ID (SQL)

An error can be returned if an invalid work area handle is passed to the function, or if an invalid search condition is passed.

Returned XML

```
<GMAPI call="search">
<status code="1">1</status>
</GMAPI>
```

Unlocking a Record

Syntax

```
<GMAPI call="Unlock">
<data name="Area">87675752</data>
</GMAPI>
```

The Unlock function unlocks a record previously locked by a call to either Append or Replace. GoldMine does not specifically release a lock on a record until you call Unlock, allowing you to perform multiple field replacements quickly. Before using Unlock, you must open a data file using the Open function.

After calling Unlock, GoldMine will also update the remote synchronization data structures to indicate the date and time that the record was modified.

Parameters

The Unlock function accepts one parameter, Area—the work area handle of the file to close. The work area handle is returned by the Open file when the file is opened.

Return Value

The Unlock function returns 1 if the record was unlocked, or 0 if an invalid work area handle was passed to the function.

Returned XML

```
<GMAPI call="Unlock">
<status code="1">Success</status>
</GMAPI>
```

Accessing Contact Records

For specific applications that need access to the GoldMine contact database at the logical level, the RecordObj function is the preferred access method. Unlike the low-level GoldMine.UI functions, the RecordObj function maintains all of the relationships between the various GoldMine files. This access method is most often used for document merging functions such as word processor mail merges or placing information into a spreadsheet.

Linking GoldMine Fields with an External Application

Syntax	`<GMAPI call="RecordObj">`
	`<data name="Command">skip</data>`
	`<data name="Argument">3</data>`
	`</GMAPI>`

The RecordObj function is a specialized function designed to link fields in a document application, such as a word processor or spreadsheet. Using RecordObj, an application can access the contact record in a high-level fashion, rather than opening the `CONTACT1.DBF` and `CONTACT2.DBF` files using Open.

Calling RecordObj within a program is equivalent to viewing and manipulating the contact record within GoldMine. The calling program can control the record pointer in the contact record much the same way a GoldMine user can move the record pointer. In fact, RecordObj can be called in such a way as to create a minimized contact record in the GoldMine work area display.

The primary differences between using Open, Move, and Read to access contact information and using RecordObj are described in the following table.

Differences in Accessing Contact Information

Using Open, Move, Read	Using RecordObj
Any filter or group that is active on a contact record in GoldMine is ignored when files are accessed using Open and Move	RecordObj can work in conjunction with a filter or group. Any records that do not match the filter expression, or are not members of the group, are skipped
The only way to maintain the relationship between the CONTACT1 and CONTACT2 files, is to manually reposition CONTACT2 whenever the record pointer is moved in `CONTACT1.DBF`.	Automatically maintains the relationship between CONTACT1 and CONTACT2 , and other contact information such as history.
	RecordObj does not contain a method to read specific fields from the database. It is expected that the application will use the Macro or Expr functions to query information from the current contact record, and use RecordObj function calls only to position the record pointer.
	When RecordObj is used to move the record pointer, the contact record screen in GoldMine is updated. To receive notification that the screen has changed, use the GoldMine.RecordObj class to receive events notifying of a record change, a tab clicked, or a contact1 or contact2 field being changed.

Parameters

The RecordObj function requires either one or two parameters.

Command: the name of the RecordObj subfunction that you want to perform.

Argument: Depending on the subfunction, a second parameter can be required. The following table lists the RecordObj subfunctions and the requirements of the second parameter.

Valid RecordObj Functions

Subfunction	Description	Argument
SETOBJECT	Create or select contact record	Optional object pointer
TOP	Move to first logical record	Not required
BOTTOM	Move to last logical record	Not required
SKIP	Skip records	Optional, recs to skip
SEEK	Seek a specific record by key	Search key value

SETORDER	Select an index	Index tag number
GETORDER	Return the currently active index name	Not required
SETTITLE	Set the contact record title	Text of title
CLOSEWINDOW	Close the contact record	None
SETRECORD	Change the behavior of SKIP, TOP, and bottom	Name of data structure to be queried
REFRESH	Repaint the contact record	Not required
GETRP	Return the point to the current contact record (Xbase) or the record ID (SQL)	Not required
GETFILTEREXPR	Get the activated filter's expression	Not required
GETGROUPNO	Get the GroupNo of the activated group	Not required
GOTO	Seeks a specific record by RecordID	The RecID to seek Additionally, accepts a third optional parameter, SetPrimary, indicating if only primary contacts should be searched (1) or (0) to include additional contacts in the search scope.

Setobject	If SetObject is called without a second parameter, subsequent calls to RecordObj will manipulate the currently active contact record. If SetObject is called with a second parameter of 0, GoldMine will create a minimized contact record in the work area display, and subsequent calls to RecordObj will manipulate that contact record. If SetObject is called with a second parameter of 1, GoldMine will create a minimized contact record in the work area display and copy any filter or group active on the last used contact record into the newly minimized contact record. If RecordObj is called with a specific pointer number, GoldMine will attempt to establish a link with that contact record.
Top	Positions the record pointer at the first logical record according to the current index order. For example, if the contact record index order is set to *Company*, a call to Top will result in the record pointer being positioned at a record with a company name such as "AAA Cleaners." GoldMine will also update the contact record to display the new record.
Bottom	Positions the record pointer at the last logical record according to the current index order. For example, if the contact record index order is set to *Company*, a call to Bottom will result in the record pointer being positioned at a record with a company name such as "Z-best Bakery." GoldMine will also display the new record.

Skip	The Skip subfunction moves the record pointer on a record-by-record basis. If Skip is called without the second parameter, it will move the record pointer to the next logical record according to the current index order. If Skip is called with a string numeric as the second parameter, the record pointer will be moved forward by the indicated number of records if the value is positive, or backwards if the value is negative. GoldMine will also update the display to show the new record. The Skip subfunction is sensitive to any filter or group that can be active on the contact record in GoldMine. For example, if the user applies a filter to the contact record in GoldMine, the Skip subfunction will skip over any records that do not match the filter expression.

Goto	The Goto subfunction positions the record pointer at the record number specified by a string numeric passed as the second parameter. Additionally, accepts a third optional parameter, SetPrimary, indicating if only primary contacts should be searched (1) or (0 - default) to include additional contacts in the search scope.
	
	`<GMAPI call="RecordObj">`
	`<data name="Command">skip</data>`
	`<data name="Argument">3</data>`
	`<data name=”SetPrimary”>1</data>`
	`</GMAPI>`

Seek	Attempts to locate a record in the data file with an index key that matches the string passed as the second parameter. Partial key searches are allowed, and GoldMine will position the record pointer at the record with the key that most closely matches the passed value. GoldMine will update the display to show the new record.
Setorder	Selects an active index for ordering and SEEKing the contact database. Only the twelve CONTACT1 indexes can be used for this subfunction. See for the appropriate values and collating sequence for each data file’s indexes.
Getorder	Returns the active index being used to sort the contact records. See for the appropriate values and collating sequence for each data file’s indexes.
Settitle	Changes both the text in the title bar of the contact record’s window and the text displayed below a minimized contact record. For example, an application that merges contact records within a document can modify the contact record title to indicate the number of records that have been merged. Any text that is passed as the second parameter will be used as the new title’s text.
Closewindow	Closes the contact record when processing is complete. Issuing this call is equivalent to selecting *Close* from the contact record’s system menu.

Setrecord	Changes the behavior of the Skip, Top, and Bottom subfunctions to allow ancillary contact information (such as additional contacts) to be queried using the RecordObj function. Normally, GoldMine assumes the CONTACT1 data file to be the parent data file, and when the Skip, Top, or Bottom subfunction is called, the record pointer is repositioned in this data file. When accessing information in GoldMine tabs, however, the Skip, Top, and Bottom subfunctions must be able to reposition the record pointer in the data file that stores these items (CONTSUPP). The SetRecord subfunction accepts the name of the data structure being queried as the second parameter. Valid data structure names are listed in the following table. ***Data Structure Name*** — ***Description*** ❍ CONTACTS — Additional contacts ❍ PROFILE — Profile records ❍ REFERRALS — Referral records ❍ LINKS — Linked documents ❍ PRIMARY — Primary contacts *Setrecord Valid Structure Names* Using SetRecord changes the behavior of the Skip, Top, and Bottom subfunctions. The first parameter is the name of the RecordObj subfunction that you want to perform. When Top is called, GoldMine will position the record pointer in the supplementary data file so that the first record containing the selected information is the current record. For example, if SetRecord is used to select CONTACTS, Top will position the record pointer on the first additional contact record for the current contact. The record pointer in the primary information data file (CONTACT1) will not be moved, so the name of the current company will remain the same. Bottom behaves in a similar manner. Skip will position the record pointer in the supplementary file on the next record of the selected type. For example, if SetRecord is used to select CONTACTS, Skip will position the record pointer in the supplementary file on the next additional contact record for the current contact. The record pointer in the primary information data file (CONTACT1) will not be moved, unless the record pointer in the supplementary file was already positioned at the last record of the selected type; then GoldMine will reposition the record pointer in the primary information data file (CONTACT1) to the next contact record and reset the record pointer in the supplementary file to the first supplemental record of the selected type. Macro expressions are also sensitive to the setting of the SetRecord subfunction.
Refresh	Repaints the contact record
GetRP	Obtains a pointer of the currently selected contact record
GetGroupNo	Returns the group number (if a group is activated)

GetFilterExpr	Returns the filter expression (if a filter is activated)

Return Value

All RecordObj subfunctions return 1 if the function was completed successfully, or 0 if an internal error occurred.

Returned XML

```
<GMAPI call="RecordObj">
<status code="1">Skip Success</status>
</GMAPI>
```

Accessing Specialized GoldMine.UI Functions

GoldMine provides a set of specialized functions for performing specific tasks, such as retrieving a list of plug-ins, adding document links to the contact database, or sending GoldMine a CallerID message.

Retrieving a List of Active Plug-Ins (GoldMine 7.0 or higher)

Syntax	<GMAPI call="GetActivatedPlugIns"/>

The GetActivatedPlugIns function is used to retrieve a list of active (trusted) plug-ins for the current user's session. For more information about GoldMine Plug-ins, see the Working with GoldMine Plug-ins chapter.

Each PlugIn node in the list is an encoded representation of the item. These are dynamically created and will not be the same starting number on individual systems. For example, 3013__GMAIL may be 3001__GMAIL on another system. The text after the number will be the same.

Each plug-in list item contains the following information:

```
XXXX__InternalName__MethodMenuEntry
```

Returned XML

```
<GMAPI call="GetActivatedPlugIns">
<status code="1">Success</status>
<data name="PlugInList">
<data name="PlugIn">3007__FrontRangeCTestControl</data>
<data name="PlugIn">3002__FrontRangeOutlookWebAccess</data>
<data name="PlugIn">3250__FrontRangeMovieViewer10__
LaunchMovieViewer10</data>
<data name="PlugIn">3251__FrontRangeMovieViewer10__
ConfigureMovieViewer10</data>
<data name="PlugIn">3001__FrontRangeTestCalendar</data>
<data name="PlugIn">3003__FrontRangeHelpAbout</data>
<data name="PlugIn">3008__GamesKittenGame</data>
<data name="PlugIn">3013__GMAIL</data>
<data name="PlugIn">3005__GoogleGoogleMaps</data>
<data name="PlugIn">3000__JCSFlashandGMViaVBNET</data>
<data name="PlugIn">3009__JCSOfficeDocument</data>
<data name="PlugIn">3004__SolutionSellingSolutionSelling</data>
</data>
```

```
</GMAPI>
```

Running a Plug-In (GoldMine 7.0 or higher)

Syntax

```
<GMAPI call="RunPlugIn">3013__GMAIL</GMAPI>
```

Or

```
<GMAPI call="RunPlugIn">3013</GMAPI>
```

Or

```
<GMAPI call="RunPlugIn">
<data name="PlugIn">3013__GMAIL</data>
</GMAPI>
```

Or

```
<GMAPI call="RunPlugIn">
<data name="PlugIn">3013</data>
</GMAPI>
```

The RunPlugIn function attempts to start the designated plug-in. For more information about GoldMine Plug-ins, see .

Returned XML

```
<GMAPI call="RunPlugIn">
<status code="1">The plug-in call was successful.</status>
</GMAPI>
```

Or

```
<GMAPI call="RunPlugIn">
<status code="0"> The Plug-in ID is invalid</status>
</GMAPI>
```

Retrieving Login Credentials for Use with the GMXS32.DLL

Syntax

```
<GMAPI call="GetLoginCredentials"/>
```

The GetLoginCredentials function is used to retrieve a string containing login credentials to be used for logging into the GMXS32.DLL through the GMW_LoadAPI, GMW_LoadBDE or GMW_Login functions. Using this option, it is not necessary to prompt the integration user for login information if GoldMine is running. The login credentials received are only valid for 30 seconds, so do not store them and attempt to use them at a later time. The string returned by this command should be used as the password to the appropriate login function, where the username is "*DDE_LOGIN_CREDENTIALS*".

Returned XML

```
<GMAPI call="GetLoginCredentials">
<status code="1">KEVIN
01C4D24F7051B9B04F882C36294F1F4AB4E4D20FCF3C1682</status>
</GMAPI>
```

Retrieving the RecID of the Current Opportunity

Syntax	<GMAPI call="GetActiveOppty"/>

The GetActiveOppty function is used to retrieve the RecID of the currently selected Opportunity in the Opportunity Manager.

Return Value

The GetActiveOppty function returns the record ID of the currently selected opportunity. If no opportunity is available, an empty string is returned.

Returned XML

No opportunity or project selected in GoldMine:

```
<GMAPI call="GetActiveOppty">
<status code="1"></status>
</GMAPI>
```

An opportunity or project is selected in GoldMine:

```
<GMAPI call="GetActiveOppty">
<status code="1">AOA73CU%Y/HD3\T</status>
</GMAPI>
```

Completing a Calendar Activity

Syntax

```
<GMAPI call="CalComplete">
<data name="Recno">ASSAG6C(+.E%3\T</data>
<data name="Activity">BIL</data>
<data name="Ref">Called Angel re Support</data>
<data name="ResultCode">DON</data>
<data name="Notes">Agreed on terms</data>
<data name="User">KEVIN</data>
 <data name=”RetainDate”>1</data>
</GMAPI>
```

The CalComplete function is used to complete an activity from the Calendar.

Parameters

The CalComplete function takes up to seven parameters.

Recno: the record number of the calendar activity to be completed.

Activity: the Activity Code. This parameter is optional.

ResultCode: the Result Code. This parameter is optional.

User: the User. If this parameter is not specified, the User field defaults to the currently logged user.

Ref: the history Reference. This parameter is optional.

Notes: the Notes for the history record. This parameter is optional.

RetainDate: a Boolean (1=true, 0= false) that if true, retains the original date of the calendar entry, otherwise uses today. Defaults to 0, false.

Return Value

The CalComplete function returns the record number (Xbase) or record ID (SQL) of the new history record created.

Returned XML

```
<GMAPI call="CalComplete">
<status code="1">1980</status>
</GMAPI>
```

Displaying Edit Windows for Calendar and History Items

Syntax	`<GMAPI call="PopCalHistItem">`
	`<data name="recID">BNPKDFZ$OF9-]WV</data>`
	`</GMAPI>`

Use the PopCalHistItem function to display the edit window for calendar or history items, including email. When you pass it a valid cal table or conthist recID, the correct edit window will open.

The Calendar Item edit window is a modal dialog: the return value will not be sent until the user closes the edit window.

For history items, the record object will align to the owner of the history automatically. This will not occur for calendar items.

General Messages

```
<GMAPI call="PopCalHistItem"><status code="-33001">
PopCalItem has failed because the passed record could not be found.
</status></GMAPI>
<GMAPI call="PopCalHistItem"><status code="-33002">
PopCalItem opens a calendar or contact history record for editing.
Parameters
RecID: the record id of the cal or conthist table entry. </status></GMAPI>
```

Return Value

- *Calendar Item Return Values*

```
<GMAPI call="PopCalHistItem"><status code="0">User pressed cancel
button.</status></GMAPI>
<GMAPI call="PopCalHistItem"><status code="1">User pressed OK
button.</status></GMAPI>
```

- *History Item Return Values*

```
<GMAPI call="PopCalHistItem"><status code="0">Failure</status></GMAPI>
<GMAPI call="PopCalHistItem"><status code="1">Success</status></GMAPI>
```

- *Email Item Return Values*

```
<GMAPI call="PopCalHistItem"><status code="0">Failure</status></GMAPI>
```

```
<GMAPI call="PopCalHistItem"><status code="1">Success</status></GMAPI>
<GMAPI call="PopCalHistItem"><status code="1">Already Open</status></GMAPI>
```

Displaying the Contact Record of an Incoming Caller

Syntax

```
<GMAPI call="CallerID">
<data name="Phone">(800)776-7889</data>
<data name="Description">Incoming caller:</data>
<data name="DisplayDialog">6</data>
 <data name="All">1</data>
<data name="UPhone">1</data>
</GMAPI>
```

The CallerID function is used to inform the GoldMine user that an incoming call has been identified by Automatic Number Identification (ANI) equipment attached to the telephone system. By using CallerID, GoldMine can perform a lookup on the contact database, and attempt to locate a contact record with a telephone number that matches the telephone number extracted by the ANI device.

With the CallerID function, GoldMine can automatically display the contact record of the caller. A dialog box is displayed, allowing the user to select an action. A CallerID function parameter is used to specify the message in the dialog box.

Parameters

The CallerID function accepts five parameters:

Phone: the telephone number of the caller as captured by the ANI device. The calling application is responsible for formatting the telephone number that appears in the Phone1 field in GoldMine.

Description: the optional message to be displayed in the dialog box in GoldMine.

All: Indicates for GoldMine to search all of the phone fields on the contact record (except FAX). Set to 1 to search all phone fields, 0 to indicate to search only Phone1.

UPhone: Indicates for GoldMine to search the UPhone fields in contact2. This parameter is ignored if the *All* parameter is set to 0.

DisplayDialog: specifies whether the dialog box is displayed. This parameter is the sum of the required options. For example, to display the caller's contact record in the current window if the record is found, or to display the contact listing if the caller's phone number is not found, specify 6 (2+4) as the <display dialog> parameter. The following table lists valid parameter values.

CallerID Parameters

Value	Description
0	Dialog box is displayed (default when third parameter is not passed)
1	Dialog box is not displayed, and contact record is displayed in a new contact record
2	Dialog box is not displayed, and contact record is displayed in the current contact record

4	Contact Listing is displayed when GoldMine cannot find the contact's telephone number. To activate this option, add this value to the third parameter value.
8	Restores input focus to the window that had input focus just before CALLERID is called—used by applications that control the entire interface.

Return Value

CallerID Return Values

Return	Description
0	Error occurred
1	Contact record found
2	Contact record not found

Returned XML

```
<GMAPI call="CallerID">
<status code="1">Passed caller was found</status>
</GMAPI>
```

Running a Counter

Syntax

```
<GMAPI call="F2Counter">
<data name="Name">My counter</data>
<data name="Inc">1</data>
<data name="Start">0</data>
<data name="Action">0</data>
</GMAPI>
```

The F2Counter function returns a sequence of consecutive numbers each time the expression is evaluated. The DDE equivalent to this function was called "Counter".

Parameters

The counter name must be unique, and can be a maximum of 10 characters. Each evaluation of the Counter function increments the counter by the Inc value.

The Start and Action parameters are optional. When Action is 1, the start value resets the counter. When Action is 2, the counter is deleted. F2Counter stores the count value between GoldMine sessions, and it is shared by all GoldMine users.

GoldMine can track an unlimited number of uniquely named counters. The counter values are stored in the LOOKUP table.

Return Value

The F2Counter function returns a number incremented by *Inc*.

Example

The following sets up the counter:

```
<GMAPI call="F2Counter">
<data name="Name">Num Iterations</data>
<data name="Inc">1</data>
<data name="Start">0</data>
<data name="Action">0</data>
</GMAPI>
```

Returns:

```
<GMAPI call="F2Counter">
<status code="1">0</status>
</GMAPI>
```

To increment the “Num Iterations” counter:

```
<GMAPI call="F2Counter">
<data name="Name">Num Iterations</data>
<data name="Include">1</data>
</GMAPI>
```

Returns:

```
<GMAPI call="F2Counter">
<status code="1">1</status>
</GMAPI>
```

Returning GoldMine Record Data

Syntax	
Range	<GMAPI call=”DataStream”> <data name=”Command”>Range</data> <data name=”Table”>Contact1</data> <data name=”Tag”>CONTNAME</data> <data name=”BotLimit”>A</data> <data name=”TopLimit”>ZZ</data> <data name=”Fields”>contact;company</data> <data name=”Filter”>*EXPRESSION*</data><! –NOT REQUIRED- > </GMAPI>
Query	<GMAPI call=”DataStream”> <data name=”Command”>Query</data> <data name=”SQL”>select recid from contact1</data> <data name=”Filter”>*EXPRESSION*</data><! –NOT REQUIRED- > </GMAPI>

Fetch	<GMAPI call="DataStream"> <data name="Command">Fetch</data> <data name="Area">1</data> <data name="FetchCount">55</data> <data name="Raw">0</data><! –NOT REQUIRED- > <data name="FieldDelimiter">|</data><! –NOT REQUIRED- > <data name="RowDelimiter">\-/</data><! –NOT REQUIRED- > </GMAPI>
Close	<GMAPI call="DataStream"> <data name="Command">Close</data> <data name="Area">1</data> </GMAPI>

DataStream returns the data of requested records from any GoldMine table using the most efficient method possible. The caller can specify the fields and expressions to return, as well as the range of records to return. A filter can optionally be applied to the data set.

The DataStream method allows for many useful applications. One example would be to publish the contents of GoldMine data on the Internet by using XSL templates with the data returned by DataStream. Web pages can be created to display GoldMine data requested by a visitor. Based on the visitor's selections, a company could dynamically present a variety of HTML pages, such as:

- Addresses of product dealers in a particular city
- Financial numbers stored in Contact2
- Seating availability of upcoming conferences

With a fast Internet connection and a strong SQL server, the GoldMine client could simultaneously respond to dozens of requests.

Record Selection

The DataStream command consists of four subcommands. Each subcommand takes different parameters.

The "range" or "query" subcommands must be called first to request the data. The "range" and "query" subcommands return an integer handle, which must be passed to the "fetch" and "close" subcommands. You must use either "range" or "query"—not both.

Datastream Range Parameters

The Table, Tag, TopLimit, and BotLimit parameters determine the range of records to scan. The Fields parameter specifies the requested fields and expression to return.

The Field parameter passed to the "range" subcommand should consist of the field names and Xbase expressions to evaluate against each record in the data set. Each field must be terminated with the semicolon (;) character. Xbase expressions must be prefixed with the ampersand (&) character and terminated with a semicolon.

The other "range" parameters are optional.

Datastream Query Parameters

The "query" subcommand sends the SQL query for evaluation on the server.

The SQL query can join multiple tables and return any number of fields. The optional Filter parameter can specify a Boolean Xbase filter expression to apply to the data set (even on SQL tables).

Datastream Fetch Parameters

The "fetch" subcommand returns a single packet string that contains the requested data from all records processed by the current "fetch" command, as specified by the second Records parameter. Optionally, Fetch can return the requested data formatted in XML, making it easy to retrieve specific data without having to parse a large string. To receive the Fetch results formatted for XML, set the "Raw" parameter to 0. Area must be the value returned from "range" or "query." The "fetch" command can be issued multiple times. The optional FieldDelimiter and RowDelimiter can override the return packet's default field and record delimiters of CR and LF. These parameters are not used when retrieving the return packet in XML format. See "Return Packet" below.

Datastream Close Parameters

The "close" subcommand must be called when the operation is complete. Unclosed data streams will leak memory and leave the database connections needlessly open. Passing an Area of 0 closes all open DataStream objects.

The XML Return Packet

DS_Fetch has an option in the GoldMine XML API to return the data in an XML format that is easier to process than the traditional datastream return packet. Consider the following DS_Query XML call:

```
<GMAPI call="DS_Query" SessionID="1">
<data name="SQL">select contact, company, key1 from contact1 where
contact='Rafael Zimberoff'</data>
<data name="Filter"/>
</GMAPI>
```

Returns

```
<GMAPI SessionID="1" call="DS_Query"><status code="1">1</status></GMAPI>
```

The DS_Fetch call to retrieve the requested data is:

```
<GMAPI call="DS_Fetch" SessionID="1">
<data name="Area">1</data>
<data name="Raw">0</data>
<data name="RecordCount">25</data>
</GMAPI>
```

The resulting XML datastream return packet is:

```
<GMAPI call="DS_Fetch">
<status code="1">Success</status>
<data name="Return">
<data name="Header">
<data name="field">
<data name="Field_Name">CONTACT</data>
<data name="Field_Type">C</data>
<data name="Field_Length">40</data>
```

```
<data name="Field_Decimal">0</data>
</data>
<data name="field">
<data name="Field_Name">COMPANY</data>
<data name="Field_Type">C</data>
<data name="Field_Length">40</data>
<data name="Field_Decimal">0</data>
</data>
<data name="field">
<data name="Field_Name">KEY1</data>
<data name="Field_Type">C</data>
<data name="Field_Length">20</data>
<data name="Field_Decimal">0</data>
</data>
</data>
<data name="CountData">3000-0001</data>
<data name="Rows">
<data Name="Row">
<data name="CONTACT">Rafael Zimberoff</data>
<data name="COMPANY">Z-Firm LLC</data>
<data name="KEY1">Partner</data>
</data>
</data>
</data>
</GMAPI>
```


The Header node contains child nodes for each field included in the SQL query, describing the fields' properties. The CountData node's text corresponds with the old fetch return packet's header data:

The first digit can be 0, 3, or 4:

0 indicates that more records are available, which could be fetched with another DS_Fetch call

3 indicates the end-of-file (EOF)

4 indicates the beginning-of-file (BOF)

Number following the dash indicates the total number of data records contained in the packet.

The Rows node contains a child node for each data record returned by the query.

Return Packet

The "fetch" command returns a single packet string containing the data from all requested records. The packet includes a header record, followed by one record for each record evaluated by "fetch." Within each record in the packet, the fields are separated by a Field Delimiter, the carriage return character by default (13 or 0x0D). The records in the packet are separated by the Record Delimiter, the line feed character by default (10 or 0x0A). These delimiters are convenient when the requested data does not contain notes from blob fields. Otherwise, you must override the default delimiters by passing other delimiter values to the "fetch" command. The characters 1 and 2 would probably make good delimiters for packets with notes.

An example of a packet of data:

```
3000-0004
Boston|23
```

```
London|393
Los Angeles|633
New York|29
```

The packet header record consists of two sections. The first byte can be 0, 3 or 4. Zero indicates that more records are available, which could be fetched with another “fetch” command. A value of 3 indicates the end-of-file (EOF), and 4 indicates the beginning-of-file (BOF). The number following the dash indicates the total number of data records contained in the packet.

Packets should be designed to be 8K to 32K. DataStream takes about as much time to read three records as it does to read 30. For best performance, adjust the number to records requested by the “fetch” command to return packets of 8K to 32K.

Performance

DataStream is the fastest way to read data from GoldMine tables. Used correctly, the GoldMine DataStream will return the data faster than most development environments would directly. DataStream offers the following advantages:

1. DataStream issues a single, efficient SQL query or Xbase seek to retrieve the records from the back-end database to the local client. On SQL databases, requests of a few hundred records could be sent from the server to the client with a single network transaction, thereby minimizing network traffic.
2. All fields and expressions are parsed initially by the “range” and “query” commands, then quickly evaluated against each record in the “fetch” command. Other lower level GoldMine.UI methods (and development environments) require that each field be parsed and evaluated each time the field’s data is read. This can save a significant amount of time when reading hundreds or thousands of records.
3. Only three calls are required to read all the data. Using traditional record-by-record querying would require one call for each field of each record (reading 10 fields from 50 records would require 500 calls).

The “range” and “query” commands execute equally fast on SQL databases. The “range” command executes much faster on Xbase tables than the “query” command.

Processing a Web Import Instruction File

Syntax

```
<GMAPI call="ExecIniImp">c:\theimport.ini</GMAPI>
OR
<GMAPI call="ExecIniImp">
 <data name="IniFile">c:\theimport.ini</data>
</GMAPI>
```

An application can send GoldMine a command to process a Web import instruction file. To start processing an instruction file, send the ExecIniImp command.

> **TIP:** For details about setting up and working with the GoldMine Web Import Gateway, see “Capturing Web Data” in Maintaining GoldMine.

Reading an Xbase Expression Without Opening a File

Syntax

```
<GMAPI call="Expr">Accountno</GMAPI>
```

OR

```
<GMAPI call="Expr">
<data name="Expression">Accountno</data>
</GMAPI>
```

The Expr function is similar to the Read function in that it attempts to evaluate an Xbase expression and return the result. The Expr function, however, does not require you to open a specific data file using the Open function. The expression passed to the Expr function is evaluated against the current operating state of GoldMine (usually, the currently displayed record), rather than the state of a specific work area. For this reason, you should be aware that differences between the return values could exist for the same expression passed to Read and Expr.

Parameters

The Expr function takes one parameter, Expression—the Xbase expression to be evaluated. GoldMine supports a subset of the Xbase dialect, so there is substantial flexibility in the application of this function.

When referencing field names within an expression, you should always use an alias; otherwise, GoldMine assumes CONTACT1 to be the default alias.

Return Value

The Expr function returns a character string containing the value of the specified expression. If an error occurs, or the expression could not be evaluated, the Expr function will return a null string.

The following XML:

```
<GMAPI call="Expr">
<data name="Expression">&amp;CityStateZip</data>
</GMAPI>
```

Returns:

```
<GMAPI call="Expr">
<status code="1">Colorado Springs, CO 80920</status>
</GMAPI>
```

Adding Merge Fields to a Form

Syntax

```
<GMAPI call="FormAddFields">
<data name="FormNo">1</data>
<data name="FieldList">contact;company</data>
</GMAPI>
```

The FormAddFields function adds merge fields to a form profile.

Parameters

The FormAddFields function takes two parameters.

FormNo: the number of the form.

FieldList: a string that lists fields, macros, and expressions; each item in the string is separated by a semicolon (;). GoldMine parses the string, checks for duplication, assigns names to the fields, and then stores the items.

Deleting Fields from a Form

Syntax	`<GMAPI call="FormClearFields">`
	`<data name="FormNo">1</data>`
	`</GMAPI>`

The FormClearFields function opens an existing form profile and deletes all associated fields.

Parameters

The FormClearFields function takes one parameter, FormNo—the number of the form.

Return Value

The FormClearFields function returns 1 if the profile is open, or 0 if an error occurs.

Closing a Form Profile

Syntax	`<GMAPI call="FormCloseForm"/>`

The FormCloseForm function closes an open form profile.

Parameters

The FormCloseForm function does not accept any parameters.

Creating an Xbase File with Registered Fields

Syntax	`<GMAPI call="FormCreateFile">`
	`<data name="FormNo">1</data>`
	`<data name="File">c:\XXXX.dbf</data>`
	`<data name="MergeCode">Mergecode</data>`
	`<data name="WhichRec">1</data>`
	`</GMAPI>`

The FormCreateFile function creates an Xbase (DBF) file with all registered fields. Any active filter or group that applies to the contact record is taken into account. FormCreateFile can be used to export data via the COM Server.

Parameters

The FormCreateFile function takes four parameters.

FormNo: the number of the form.

File: the name of the .DBF file to be created.

MergeCode: the merge code. If any merge code value(s) are included in the function, only records with the matching merge code(s) will be included. To include multiple merge codes, place a space between each individual merge code. If the MergeCode parameter is empty, all records are included.

WhichRec: indicates which records are to be exported. The WhichRec value is the sum of values for each available listed below.

WhichRec Values

Value	Description
1	Primary
2	Secondary
4	All records
8	Forward to last
16	Return control to the calling program immediately after export has started

Examples of WhichRec Parameter

Current contact	1
All primary contacts	5 (1+4)
Forward to last of primary and additional contacts	11 (1+2+8)

Return Value

The FORMCREATEFILE function returns the total number of records in the output .DBF file.

Returning a Field Name for an Expression

Syntax

```
<GMAPI call="FormGetFieldName">
<data name="FormNo">1</data>
<data name="Field">contact</data>
</GMAPI>
```

The FormGetFieldName function returns the field name for an expression, a macro, or a field.

Parameters

The FormGetFieldName function takes two parameters.

FormNo: the number of the form.

Field: the name of the field, macro, or expression to be associated with the file name.

Returning a Value for Unattached Fields

Syntax

```
<GMAPI call="FormNewFormNo"/>
```

Return Value

The FormNewFormNo function returns a new, unique FormNo value that can be used to register fields not attached to a GoldMine form.

Counting the Number of Exported Records

Syntax	`<GMAPI call="FormQueryCreate">`
	`<data name="Flags">0</data>`
	`</GMAPI`

The FormQueryCreate function provides status information during an export by returning the number of records exported during the export process.

Parameters

The FormQueryCreate function takes one optional parameter, Flags.

The following table lists values of FormQueryCreate parameters.

FormQueryCreate Parameters

Value	Description
0	Export in progress (default)
1	Start process
2	Abort process

Return Value

The FormQueryCreate function returns the number of records created while an export is in progress, or -1 when the record export process is completed.

FormPrintedDoc

Syntax	`<GMAPI call="FormPrintedDoc">`
	`<data name="RecordID"> 9NDJRJN(EQ[)JW:</data>`
	`</GMAPI`

The FormPrintedDoc function is used to complete a pending literature fulfillment request. Call this function after printing the merge form to remove the pending literature fulfillment and create a history record.

Parameters

RecordID: the RecID of the pending literature fulfillment request.

Creating a History Record

Syntax

```
<GMAPI call="InsHist">
<data name="AccNo">A3042474804 WB9!JCat</data>
<data name="Activity">SLS</data>
<data name="Duration">00:35:00</data>
<data name=”OpRecID”>ValidOpRecid</data>
<data name="RecType">C</data>
<data name="Ref">Informed Paul of sale terms</data>
<data name="ResultCode">DON</data>
<data name="Notes">Ready to proceed to next step</data>
<data name="User">KEVIN</data>
<data name="Private">1</data>
</GMAPI>
```

The InsHistory function is used to create a history record in GoldMine. The InsHistory function provides a higher level interface for creating these records than using Open, Append, and Replace.

Parameters

AccNo: the account number of the contact record to which the new history record will be linked.

Rectype: the record type to create. The following values are available:

InsHistory Activity Valid Values

Value	Record Type	Value	Record Type
A	Appointment	U	Unknown
C	Phone call	CC	Call back
D	To-do	CI	Incoming call
E	Event	CM	Returned message
L	Form	CO	Outgoing call
M	Sent message	MG	E-mail message
O	Other	MI	Received e-mail
S	Sale	MO	Sent e-mail
T	Next action		

Duration: the length of time spent on the activity. Format as HH:MM:SS. (optional)

OpRecid: the Recid of the opportunity or project record to link the history activity.
Omit if not linking to a project or opportunity (optional).

Ref: the history reference.

Notes: the Notes for the history record (optional).

Activity: the Activity Code (optional).

ResultCode: the Result Code (optional).

User: the User (optional). If this parameter is not specified, the User field defaults to the currently logged user.

Private: flag to specify if the history activity should be marked private. Set to 1 for private, or 0 to public.

Return Value

The InsHistory function returns the record number (Xbase) or record ID (SQL) of the new history record if the function was completed successfully. The function returns 0 if a new record could not be appended to the data file.

Returned XML

```
<GMAPI call="InsHist">
<status code="1">1982</status>
</GMAPI>
```

Creating or Updating a Document Link

Syntax

```
<GMAPI call="LinkDoc">
<data name="RecNo">0</data>
<data name="File">C:\Documents and Settings\Kevin\My
 Documents\GMAPI\TLog_Mechanics.pdf</data>
<data name="Desc">Help File</data>
<data name="User">KEVIN</data>
<data name="Notes">Read this</data>
 <data name="Sync">1</data>
</GMAPI>
```

The LinkDoc function is used to create or update a document link in GoldMine. Document links allow you to launch directly into an application and load the application with a document by clicking on the desired document listed in the contact's Links tab. GoldMine maintains these links as records in the supplementary data file. The LinkDoc function provides a higher level interface to these records than can be obtained by using Open, Append, and Replace.

Parameters

RecNo: the record number of the link record to be updated. If a new link record is to be created, pass 0 as the first parameter.

File: the fully qualified path and filename of the file to link. Keep in mind that a valid association must exist for the file's extension if GoldMine is to automatically launch the file's application.

Desc: the document title.

User: the optional document owner. If this field is not passed, the document owner defaults to the name of the currently logged GoldMine user.

Notes: optional notes for the linked document record in the Links tab.

Sync: defines the remote synchronization status for the linked document from the values shown in the following table.

Sync Valid Values

Value	Action
-1	Uses the GoldMine default as defined **by Allow new documents to sync by default** in the **Sync** tab of the **Preferences** window.
0	Does not synchronize the newly linked document.
1	Allows the newly linked document to synchronize.

Return Value

The LinkDoc function returns the new record number (Xbase) or record ID (SQL) if the function was completed successfully. The function returns any empty string if a new record could not be appended to the data file, or an existing record could not be locked for update.

Returned XML

```
<GMAPI call="LinkDoc">
<status code="1">482</status>
</GMAPI>
```

Displaying a Message Dialog Box

Syntax

```
<GMAPI call="MsgBox">
<data name="Message">Are you sure?</data>
<data name="Style">4</data>
</GMAPI>
```

The MsgBox function displays a standard Windows message dialog box.

Parameters

The MsgBox function accepts two parameters.

MsgBox: the message to display within the dialog box.

Style: the optional style of the message box. This value is the sum of the following options:

MsgBox Style Values

Value	Meaning
0	Display OK button only
1	Display OK and Cancel buttons

2	Display Abort, Retry, and Ignore buttons
3	Display Yes, No, and Cancel buttons
4	Display Yes and No buttons
5	Display Retry and Cancel buttons
16	Display Stop icon
32	Display Question Mark icon
48	Display Exclamation Mark icon
64	Display Information icon
128	First button is default
256	Second button is default
512	Third button is default

Return Value

The MsgBox function returns the following values:

MsgBox Return Values

Return	Description
1	OK button selected
2	Cancel button selected
3	Abort button selected
4	Retry button selected
5	Ignore button selected
6	Yes button selected
7	No button selected

Returned XML

```
<GMAPI call="MsgBox">
<status code="1">6</status>
</GMAPI>
```

Adding a Merge Form

Syntax

```
<GMAPI call="NewForm">
 <data name="AppType">Microsoft.Word.10</data>
 <data name="Template">c:\Program
 Files\GoldMine\Templates\Proposal.doc</data>
 <data name="Title">Business Proposal</data>
 <data name="Macro">[MsgBox("Form Added","0")]</data>
 <data name="FormType">0</data>
 <data name="Flags">3</data>
</GMAPI>
```

The NewForm function adds a merge template record into the Merge Forms window in GoldMine. This function's DDE counterpart is used primarily by the document merge link installation macro; however, the function can also be used to add additional merge templates from a user-written application.

Parameters

The NewForm function takes up to six parameters; the first three parameters are required, and the last three parameters are optional.

AppType: the type of document to which the new form record will point. This value must be a valid Application Identifier, such as Word.Document.6, that corresponds to an entry in the Registration Database.

Template: the fully qualified path and filename of the template file.

Title: the title of the document as it should appear in the Merge Forms browse window.

Macro: the name of an optional DDE function to be called after the template is loaded by the linked application. If this parameter is not specified, the default function is MAINMENU. This parameter must be passed in DDE call format.

FormType: the optional type of template. If this parameter is not specified, the template type is assumed to be Document. GoldMine accepts the following values for this parameter:

Document Types

Type	Description
0	Document
1	Spreadsheet
2	Other

Flags: a three-character field corresponding to the values of the *Link To Doc*, *Save History* and *Allow Hot Link* options on the *Form Setup* dialog box. To set (check) one of these options, 1 is passed; to reset (uncheck), 0 is passed.

Flag Values

Position	Description
0	Link To Doc check box
1	Save History check box
2	Allow Hot Link check box

Return Value

The NewForm function returns a form number.

Playing a Toolbar Macro

Syntax

```
<GMAPI call="PlayMacro">
<data name="Macro">800</data>
<data name="Wait">0</data>
</GMAPI>
```

A macro groups together a series of commands, keystrokes, and/or mouse clicks into a one-step operation. You can create a macro to automate a sequence of tasks that you perform frequently in GoldMine. This function plays a macro previously created in GoldMine.

Parameters

The PlayMacro function takes two parameters that identify the macro and assign a wait state.

Macro: The first parameter identifies the macro. Either the number for the currently logged user or a valid macro filename can be used to identify a macro.

Identifying a Macro by Number

Each user can create up to 100 macros from the GoldMine toolbar. Each macro can be assigned an optional numeric identification from 800 to 899. For example, you can assign 800 to identify your first macro, 801 to identify your second macro, and so on.

> **TIP:** For details about creating a macro from the GoldMine toolbar, see "About Macros" in the online Help.

Identifying a Macro by File Name

You can assign a file name to identify the macro, such as C:\GOLDMINE\MACROS\JOHN.801.

Wait: The second parameter assigns a wait state that determines GoldMine availability to process another macro or task while the current macro executes. To set GoldMine to wait for the currently executing macro to finish before starting another task, set the parameter to 1. For example, if you are setting up a sequence of macros to run tutorial lessons, you want GoldMine to wait for each lesson to finish before executing the next macro that will run the following lesson.

To allow GoldMine to perform background processing, such as indexing, while the macro(s) execute, set the parameter to 0.

Return Value

The PlayMacro function returns an integer value based on the wait parameter; that is, GoldMine availability to process a task in addition to the currently running macro. If the wait parameter is 0 (GoldMine does not wait for the macro to finish to process another task), the PlayMacro function will always return 1. If the wait parameter is 1 (GoldMine will wait for the current macro to finish before processing another macro or task), the PlayMacro function will return either 0 or 1 under the following conditions:

PlayMacro Return Values

Return	Description
0	Error occurred during macro playback
1	Macro played successfully

You can also play a macro from the command line (DOS prompt). Executing a macro from the command line can be useful in running functions at night, such as indexing, running an Automated Process, or synchronizing with remote sites with a transfer set created via macro. You can either identify a macro by an identification number, like GMW4 /m:801, or by file name like GMW4 /m:c: \index.801. If necessary, the command line statement can start GoldMine and then, once started, run the macro.

Optional switches include:

/m: Logs in automatically to GoldMine

/u:[username] Provides the username entry to log in to GoldMine

/p:[password] Provides the password entry to log in to GoldMine

If running the Plus! Pack with Windows, you can run a macro from the System Agent by placing a command line switch for GoldMine in the Program field of the Schedule a New Program dialog box that will run a macro. For example, to log in John with his username and password, then run John's first macro, place the following macro in the System Agent:

GMW5 /u:john /p:pswd /m:800

Where *GMW5/* starts Goldmine, u:john/ is login user John, *p:pswd/* enters the password password, and *m:800* runs first macro.

Creating and Sending a Pager Message

Syntax

```
<GMAPI call="SendPage">
<data name="Message">Your 3:00pm appointment is
cancelled</data>
<data name="To">PAULR</data>
<data name="From">Trish</data>
</GMAPI>
```

The SendPage function allows you to create and send a message to the pager of a GoldMine user. The function consists of the following components:

Message can consist of any text message that you create with this function to send to a pager; most pages can accept messages of 70–100 characters.

From includes the sender's name as an optional "signature."

To identifies an optional GoldMine user who will receive the pager message. Information about the pager must be entered in the Edit|Preferences|Pager tab, such as ID code or PIN number, telephone number of the pager, and maximum message size in characters that the pager can accept.

Return Value

The SendPage function can return one of two values.

SendPage Return Values

Return	Description
0	Error occurred during the attempt to send the message to the pager
1	Pager message was transmitted successfully

Displaying a Message in the GoldMine Status Bar

Syntax

```
<GMAPI call="StatusMsg">
<data name="Message">Waiting for command</data>
<data name="Delay"/>
</GMAPI>
```

The StatusMsg function displays a message in the GoldMine status bar.

Parameters

Message: the message to be displayed in the status bar.

Delay: an optional delay, after which time the message is removed from the status bar.

Returned XML

```
<GMAPI call="StatusMsg">
<status code="1">Success</status>
</GMAPI>
```

Converting TLog Timestamps

Syntax

```
<GMAPI call="SyncStamp">
 <data name="Stamp">20040120:10:36:52</data>
</GMAPI>
```

The SyncStamp function converts a TLog timestamp to a date and time representation, and from a date and time representation back to the TLog time stamp format.

Parameter

The SyncStamp function takes one parameter, *Stamp*.

Return Value

When the *Stamp* parameter is exactly 17 characters long, formatted as Date:Time in form of CCYYMMDD:HH:MM:SS, the return string is in TLog time stamp format, exactly seven characters long. When the *Stamp* parameter is seven characters long, and formatted as a TLog timestamp, the return string is formatted as CCYYMMDD:HH:MM:SS. An empty return string indicates an error.

Returned XML

```
<GMAPI call="SyncStamp">
<status code="1">A6P9FC8</status>
</GMAPI>
```

Updating the Sync Log File

Syntax

XML	`<GMAPI call="UpdateSyncLog" >`
	`<data name="Table">Contact1</data>`
	`<data name="RecID">9NDJRJN(EQ[)JW:</data>`
	`<data name="Field">Key3</data>`
	`<data name="Action">U</data>`
	`</GMAPI>`

Parameters

Table specifies the table name (such as “Contact1”) or the table ID.

RecID specifies the RecID of the updated record: the correct RecID must be passed, and the RecID value must be exactly 15 characters long.

Field specifies the name of the field that has changed. This parameter is only relevant when the Action parameter is U. Field is ignored when Action is N or D.

Action should be N when a new record has been appended, D when a record has been deleted, or U when a field in a record has been updated.

Return Value

The UpdateSyncLog function returns the following XML:

```
<GMAPI call="UpdateSyncLog">
<status code="4">Field TLog entry created.</status>
</GMAPI>
```

UpdateSyncLog Code Attribute Values

1. Return	*2.* Description

0	Error
1	New TLog entry created
2	New TLog entry updated
4	Field TLog entry created
8	Field TLog entry updated
16	Deleted record TLog entry created
32	New TLog Entry removed

Importing a Prepared TLog Import File

ReadImpTLog reads the status of a TLog import file, then deletes the import file when the process is completed.

Syntax

XML

```
<GMAPI call="ReadImpTLog" >
<data name="File">c:\tlogs\mytlog.dbf</data>
<data name="Delete">1</data>
</GMAPI>
```

Parameters

File specifies the import file name—see below for the import file structure.

Delete specifies to delete the import file when the process has completed.

Return Value

ReadImpTLog function returns the following values in the code attribute:

ReadImpTLog Code Attribute Values

Code	Description
0	Failure
1	Success -- Text is total number of imported TLog records

Notes

Your application can determine when the imported process completes by setting the Delete parameter to 1, and noting when the import file is deleted. The TLog import must have the structure shown in the following table.

TLog Import Structure

Field Name	Type	Length

Table ID	char	10
RecID	char	15
Field ID	char	10
Action ID	char	1

Forcing Logout

Syntax

XML

```
<GMAPI call="ForceLogout" >
<data name="LogoutSelf">1</data>
<data name="Relogin">1</data>
<data name="InMinutes">1</data>
</GMAPI>
```

The ForceLogout command forces all users to logout of GoldMine.

Parameters

LogoutSelf: specifies if the currently logged in user should also be logged out. 1 for rue, 0 for false.

Relogin: Set to 1 to indicate for GoldMine to relogin after the users are logged out.

InMinutes: Specifies the number of minutes to wait before forcing the logout.

Reading Security and Rights

Retrieving User Permissions

The UserAccess function retrieves specific permission information for the logged-in user.

Syntax

XML

```
<GMAPI call="UserAccess"/>
```

This command returns a data element for each of the following permissions for the logged in user. The text value of the data element will be either 0 or 1, indicating if the permission is granted for the user.

Permissions Returned by UserAccess

Rights
☐ Master Rights
☐ Other User Calendar Access
☐ Other User History Access
☐ Other User Sales Access

- ☐ Other User Report Access
- ☐ Other User Merge Form Acccess
- ☐ Other User Filter Access
- ☐ Other User Groups Access
- ☐ Other User Links Access
- ☐ Create Records
- ☐ Edit Records
- ☐ Delete Records
- ☐ Change Owner
- ☐ Field Views
- ☐ Schedule APs
- ☐ SQL Queries
- ☐ NetUpdate
- ☐ Build Groups

Returned XML


```
<GMAPI call="UserAccess">
<status code="1">Success.</status>
<data name="return">
<data name="Master Rights">1</data>
<data name="Other User Calendar Access">1</data>
<data name="Other User History Access">1</data>
<data name="Other User Sales Access">1</data>
<data name="Other User Report Access">1</data>
<data name="Other User Merge Form Access">1</data>
<data name="Other User Filter Access">1</data>
<data name="Other User Groups Access">1</data>
<data name="Other User Links Access">1</data>
<data name="Create Records">1</data>
<data name="Edit Records">1</data>
<data name="Delete Records">1</data>
<data name="Change Owner">1</data>
<data name="Field Views">1</data>
<data name="Schedule APs">1</data>
<data name="SQL Queries">1</data>
<data name="NetUpdate">1</data>
<data name="Build Groups">1</data>
</data>
</GMAPI>
```

Retrieving Calendar Permissions

Using CalAccess, you can query whether the user logged in to GoldMine has permissions to read/write a particular CAL record.

Syntax

XML	`<GMAPI call="CalAccess">`
	`<data name="RecordType">C</data>`
	`<data name="User">KEVIN</data>`
	`<data name="Number1">22</data>`
	`</GMAPI>`

Parameters

Pass this command the record type and number1 value from the calendar record in question. Also pass the user you wish to query if they have permission to this record or not.

RecordType is the RecType of the record.

User is the UserID of the record.

Number1 is the Number1 value of the record.

Return Value

The CalAccess function returns *1* if the user has rights to read/write.

Retrieving History Access

Using HistAccess, you can query if the user logged has rights to read/write a CONTHIST record.

Syntax

XML	`<GMAPI call="HistAccess">`
	`<data name="RecordType">C</data>`
	`<data name="User">KEVIN</data>`
	`</GMAPI>`

Parameters

Pass this command the record type value from the calendar record in question. Also pass the user you wish to query if they have permission to this record or not.

RecordType is the RecType of the record.

User is the UserID of the record.

Return Value

The HistAccess function returns *1* if the user has rights to read/write.

Macros

To facilitate the use of DDEAUTO fields, GoldMine allows you to select a macro as the service item. Upon encountering a DDE service item that starts with an ampersand (&), GoldMine searches an internal table of macro names. If a match is found, the macro is processed and the result is returned, as if a DDE function or expression had been used. The GoldMine COM Server recognizes these same macros for use in such methods as Expr and Macro.

Most macros are sensitive to the setting of the RECORDOBJ function's SETRECORD subfunction. This function is used primarily to gain access to additional contacts and other supplementary information. When the SETRECORD type is set to PRIMARY, the following macros will return the value from the corresponding fields in the primary information portion of the contact record. When the SETRECORD type is set to CONTACTS (additional contacts), or another supplementary record type, the macros will return the value from the corresponding field in the supplementary file (`CONTSUPP.DBF`).

Executing Macros

To evaluate any of the macros described in this section, use the Macro command for the GoldMine COM Server.

Syntax	`<GMAPI call="Macro">`
	`<data name="Macro">&FullAddress</data>`
	`</GMAPI>`

Returned XML

The XML returned will of course vary based on the Macro requested.

For the example in the Syntax table above, the XML returned is:

```
<GMAPI call="Macro">
<status code="1">1150 Kelly Johnson Blvd. Colorado Springs, CO 80920
</status>
</GMAPI>
```

Available Data-Related Macros

&Address	Returns a string containing the values of both *&Address1* and *&Address2*, separated by a carriage return and line feed character. If either **&Address1** or *&Address2* does not contain any data, a single line of data is returned, without the carriage return and line feed character. This macro can be used to perform rudimentary blank line suppression within linked applications that do not support blank address line suppression internally. The action of this macro string is similar to the action of the *&Address* macro. The *&Address2* macro can be used to return an additional contact address by using the RECORDOBJ SETRECORD subfunction.

&Address1	Returns the first *Address* field from the active contact record. Typically, this value will be extracted from the *Address1* field in the primary display portion of the contact record; however, when the RECORDOBJ SETRECORD subfunction has been used to change the returned record type to CONTACTS, then GoldMine returns the value from the *Address1* field on the additional contact record, if a value is entered. When the *Address1* field on the additional contact record is blank, then the *&Address1* macro returns the value in the *Address1* field in the primary display portion of the contact record. When the RECORDOBJ SETRECORD type is set to return a record type other than CONTACTS, the *&Address1* macro returns the value in *Address1* field in the primary display portion of the contact record.
&Address2	Returns the second *Address* field from the active contact record. Typically, this value will be extracted from the *Address2* field in the primary display portion of the contact record; however, when the RECORDOBJ SETRECORD subfunction has been used to change the returned record type to ADDITIONAL, then GoldMine returns the value from the *Address2* field on the additional contact record, if an entry exists in the *Address2* field on the additional contact record. When the *Address2* field on the additional contact record is blank, then the *&Address2* macro returns the value in the *Address2* field in the primary display portion of the contact record. When the RECORDOBJ SETRECORD type is set to return a record type other than PRIMARY or ADDITIONAL, the *&Address2* macro returns the value in the *Address2* field of the primary display portion of the contact record.
&BrowseRecNo	*Xbase*: Returns the record number of the last selected record in a browse window. *SQL*: Returns the record ID of the last selected record in a browse window.
&CalRefresh	Refreshes the graphical calendar display.
&City	Returns the *City* field from the active contact record. The action of this macro string is similar to the action of *&Address1*. The *&City* macro can be used to return an additional contact city by using the RECORDOBJ SETRECORD subfunction.
&CityStateZip	Returns a format string of text containing the *City*, *State*, and *Zip* fields from the active contact record. This string is returned in the following format: *City*, *State Zip* The action of this macro string is similar to the action of *&Address1*. The *&CityStateZip* macro can be used to return an additional contact city, state, and ZIP Code by using the RECORDOBJ SETRECORD subfunction.
&CommonDir	*Xbase*: Returns the path information for the directory where the contact sets are located. *SQL*: Returns the BDE alias where the contact sets are located.

&Contact	Returns a Contact name from the active contact record. Normally, this value will be extracted from the Contact field in the primary display portion of the contact record; however, the RECORDOBJ SETRECORD subfunction can be used to change the returned record type to additional contact, or another type of supplementary record. When the RECORDOBJ SETRECORD type is set to return record types other than PRIMARY, the &Contact macro returns the value in Contact field in CONTSUPP for the current supplementary record.
&Country	Returns the Country field from the active contact record. The action of this macro string is similar to the action of &Address1. The &Country macro can be used to return an additional contact country by using the RECORDOBJ SETRECORD subfunction.
&Dial1	Returns the Phone1 entry from the active contact record. The returned phone number is formatted for dialing. GoldMine applies the same rules used to dial the phone via TAPI. If selected, PREDIAL.INI settings are applied to phone number selection.
&Dial2	Returns the Phone2 entry from the active contact record. For details, see &Dial1 above.
&Dial3	Returns the Phone3 entry from the active contact record. For details, see &Dial1 above.
&DialFax	Returns the FAX entry from the active contact record. For details, see &Dial1 above.
&EmailAddress	Returns the primary e-mail address for the currently selected contact.
&Fax	Returns the fax number as it should be sent to an auto-dialer for automatic fax transmission.
&Filter	Returns the activated filter expression.
&FirstName	Returns the first name of the current contact.
&FullAddress	Returns a string containing the complete address for the contact record, composed of values of &Address1, &Address2, &City, &State, and &ZIP. The action of this macro string is similar to the action of &Address1. The &FullAddress macro can be used to return an additional contact address by using the RECORDOBJ SETRECORD subfunction.

&GetRoTabID	Returns the ID of the currently selected tab. Typically, this value will verify that the correct tab is selected when a user starts a custom application.
	
	The following values are valid:
	
	0 = Summary
	1 = Fields
	2 = GM+View
	3 = Notes
	4 = Contacts
	5 = Details
	6 = Referral
	7 = Pending
	8 = History
	9 = Links
	10 = Members
	11 = APs/Tracks
	12 = Opportunities
	13 = Projects
	14 = Relationships/Org tree
	15 = Cases
	16 = HEAT View if installed, else it will go to the first tab
	17+ = custom if installed, otherwise the first tab
	
	The following example tests the selection of the **Details** tab:
	
	`<GMAPI call="Macro">&GetROTabID</GMAPI>`
	
	Returns:
	
	`<GMAPI call="Macro"><status code="1">1</status></GMAPI>`

&GetRoTabPos	Returns the currently selected tab position. Since the tabs can be rearranged, this method is not always reliable for determining the currently selected tab. For details, see &GetRoTabID.
&GoldDir	*Xbase*: Returns path information for the directory in which GoldMine is installed. *SQL*: Returns path information for BDE alias in which GoldMine is installed.
&LastFirstName	Returns the name of the current contact in the format: last name, first name
&LicUsers	Returns the number of concurrent users allowed to log in to the installed copy of GoldMine.

&LicUsersAvailable	Returns the number of users allowed to log in to the installed copy of GoldMine license.
&NameAddress	Returns a string containing the contact's name, company, and complete address of the current contact record. Each address line is separated by a carriage return and line feed, and the entire string is formatted so that the string can be inserted directly into a merge template. If any of the address lines on the contact record is empty, that address line will be suppressed. This macro can be used to perform rudimentary blank line suppression within linked applications that do not support blank address line suppression internally. The action of this macro string is similar to the action of the &ADDRESS macros, and the &NAMEADDRESS macro can be used to return an additional contact address by using the RECORDOBJ SETRECORD subfunction.
&NameTitleAddress	Returns a string containing the contact's name, title, department, company, and complete address of the current contact record. Each line is separated by a carriage return and line feed, and the entire string is formatted so that the string can be inserted directly into a merge template. If any of the lines on the contact record is empty, that line will be suppressed. This macro can be used to perform rudimentary blank line suppression within linked applications that do not support blank address line suppression internally. The action of this macro string is similar to the action of the &ADDRESS macros, and the &NAMETITLEADDRESS macro can be used to return an additional contact address by using the RECORDOBJ SETRECORD subfunction.
&NewRecID	Returns a unique record ID, which can be used when creating new records.
&Notes	Returns the *Notes* from the active contact record. Typically, this value will be extracted from the *Notes* field in the primary display portion of the contact record; however, the RECORDOBJ SETRECORD subfunction can be used to change the returned record type to additional contact, or another type of supplementary record. When the RECORDOBJ SETRECORD type is set to other than PRIMARY, the &TITLE macro returns the value in Notes field in CONTSUPP for the current supplementary record.
&Phone	Returns a telephone number from the selected contact record. The action of this macro string is similar to the action of the &ADDRESS1. The &PHONE macro can be used to return an additional contact telephone number by using the RECORDOBJ SETRECORD subfunction.

&Profile(s)

Two related macros:

- *&Profile*: Returns the first matching profile record for the selected contact.
- *&Profiles*: Returns all profile records for the selected contact.

Both of these macros take optional parameters. Each parameter must be separated by a period (.). The following examples show the syntax for the *&Profile* and *&Profiles* macros:

&Profile Example 1

```
&Profile.ProfileName.Reference.Flags
```

Retrieves the first profile that matches the ProfileName and Reference.

The Reference parameter is optional. If passed, the Reference parameter acts as a "begin with" condition on the profile reference. If the Reference parameter is not passed, all ProfileName profiles are evaluated.

The optional Flags parameter has the following values:

2 Returns the extended profile fields

4 Returns the ProfileName and Reference

The &Profile(s) macro can easily fill in a Word table with the selected contact's profile information because tabs separate each field value, and a CR/LF separates each profile record.

&Profile Example 2

The following example returns the first e-mail address of the contact:

```
&Profile.E-mail Address
```

&Profiles Example 1

The following example returns all the computer profiles that begin with the word notebook:

```
&Profiles.Computer.Notebook
```

&Profiles Example 2

The following examples use the Flags parameter to specify the profile fields to return:

```
&Profiles.Computer.Notebook
Notebook ThinkPad 770|
Notebook Compaq Elite|
Notebook Dell 1200|
&Profiles.Computer.Notebook.2
Computer|Notebook ThinkPad 770|
Computer|Notebook Compaq Elite|
Computer|Notebook Dell 1200||
&Profiles.Computer.Notebook.4
Computer|Notebook ThinkPad 770|IBM|233Mz|
Computer|Notebook Compaq Elite|Compaq|200mz|
Computer|Notebook Dell 1200|Dell|166mz|
```

&ProgramDataDir	Returns the place where the GM.ini, user.ini, and anything that needs to have read/write access in GoldMine can be found. It is very similar to the split path installs that GoldMine had when Windows XP was released. For non-split paths, it will return the SysDir.
	
	Example:
	
	`<GMAPI call="Macro">Programdatadir</GMAPI>`
	
	Returns :
	
	`<GMAPI call="Macro"><status code="1">c:\code\GMDev8.0_Main\bin\debug\</status></GMAPI>`

&RoTabPage	Returns the currently selected tab. Typically, this value will verify that the correct tab is selected when a user starts a custom application. Values between 1 and 9 represent tabs in the first row of tabs; for example, 1 represents the *Summary* tab. Values between 10 and 18 represent tabs in the second row, and 19–27 represent tabs in the third row.
&SerialNo	Returns the serial number of the installed GoldMine program.

&SetRoTab#	Selects the tab that corresponds to the number (represented by #) in the active contact record. The following values are valid: 1 = Summary 2 = Fields 3 = GM+View 4 = Notes 5 = Contacts 6 = Details 7 = Referral 8 = Pending 9 = History 10 = Links 11 = Members 12 = APs/Tracks 13 = Opportunities 14 = Projects 15 = Relationships/Org tree 16 = Cases 17 = HEAT View if installed, else it will go to the first tab 18+ = custom if installed, otherwise the first tab Example: `<GMAPI call="Macro">&SetROTab4</GMAPI>` Displays the Notes tab in the contact record.
&ShutDown	Logs out the currently logged user, and quits GoldMine.
&State	Returns the *State* field from the active contact record. The action of this macro string is similar to the action of the &ADDRESS1. The &STATE macro can be used to return an additional contact state by using the RECORDOBJ SETRECORD subfunction.
&SysDir	Returns the GoldMine system directory.
&SysInfo	Displays system information as returned by *Help > About GoldMine > System Info*.
&Title	Returns the *Title* from the active contact record. Normally, this value will be extracted from the *Title* field in the primary display portion of the contact record; however, the RECORDOBJ SETRECORD subfunction can be used to change the returned record type to additional contact, or another type of supplementary record. When the RECORDOBJ SETRECORD type is set to other than PRIMARY, the &TITLE macro returns the value in Title field in CONTSUPP for the current supplementary record.

&User_Var	Returns the defined field value from all users, a specified user, or the currently logged user. For details on defining values, see "Defining Field Values for use with External Applications" in Maintaining GoldMine.
	The &User_Var macro allows GoldMine users to store specific data that can be retrieved later into applications that are linked with GoldMine. This macro can be defined in the [user_var] section of both the GM.INI and the username.INI of GoldMine.
	Usage Syntax:
	`&User_Var.<variable name>.<GoldMine username>`
	Example:
	`&User_Var.Territory.Dan`
	(Where <variable name> is a descriptive name of the macro and <GoldMine username> assigns a defined value to a specific GoldMine user.)
	<GoldMine username> is optional, as GoldMine will assign these values to the current GoldMine user.

&UserFullName	Returns the full name of the currently logged GoldMine user as the name appears in the *FullName* field in the *Users Master File* for the user.
&UserName	Returns the login name of the currently logged GoldMine user.
&Version	Returns the version number of the installed GoldMine program.
&WebSite	Returns *http://<Web site>* for the active contact.
&ZIP	Returns the Zip field from the currently active contact record. The action of this macro string is similar to the action of the &ADDRESS1. The &ZIP macro can be used to return an additional contact ZIP Code by using the RECORDOBJ SETRECORD subfunction.

Macros for Merge Forms

The following macros are used primarily for creating links to GoldMine through the Merge Forms function. The values returned by each of these macros are updated by GoldMine when a Merge Form is launched by selecting Edit, Link, Print or Fax from the Merge Forms dialog box.

&PARAM1 (filename)	Returns the path and filename of the document template associated with the merge form selected when *Edit*, *Link*, *Print*, or *Fax* was selected. This value is obtained from the *Template File* field in the merge form's *Form Setting* dialog box.
&PARAM2 (action)	Returns a value indicating whether the *Edit*, *Link*, *Print*, or *Fax* button was selected to launch linked application.

&PARAM2 Parameters

Value	Description
1	**Edit** selected

2	**Link** selected
3	**Print** selected
4	**Fax** selected

&PARAM3 (range)	Returns a value corresponding to the setting of the **Record Range** options on the **Merge Forms** dialog box when the **Edit, Link, Print,** or **Fax** button was selected.

&PARAM3 Parameters

Value	Description
1	**This contact** selected
2	**All contacts** selected
3	**Forward to last** selected

&PARAM4 (scope)	Returns a value corresponding to the setting of the **Primary** and **Additional** check boxes on the **Merge Forms** dialog box when the **Edit, Link, Print**, or **Fax** button was selected.

&PARAM4 Parameters

Value	Description
1	**Primary** checked
2	**Additional** checked
3	Both **Primary** and **Additional** checked

&PARAM5 (flags)	Returns a value corresponding to the status of the **Link to Doc, Save History,** and/or **Allow Hot Link** check boxes on the **Merge Forms** dialog box. In addition, the returned value determines whether the form was merged as the result of an Automated Processes action. Returns a seven-character string. Each position of the string can contain either 0, indicating the item was not checked (or Automated Processes is not active), or 1, indicating the item was checked (or Automated Processes is active).

&PARAM5 Parameters

Position	Description
1	Link to Doc
2	Save History
3	Allow Hot Link
4	Unused
5	Unused
6	Unused
7	Automated Processes status

&PARAM6 (LinkDoc record number)	Returns a value containing the record number of the last Linked Document supplementary record created as a result of launching a Merge Form. When you launch a merge form with **Link to Doc** selected, GoldMine creates a linked document record to hold the saved document. This value can be saved and used to update the linked document record by passing the record number to the LinkDoc function.
&PARAM7 (contact record pointer)	Returns a pointer to a minimized contact record that is created when **Print** or **Fax** is selected on the **Merge Forms** dialog box, and the **Record Range** is **All Contacts** or **Forward to Last**. This value can then be passed to the RecordObj function to further control a document merge from the linked application.
&PARAM8 (merge code value)	Returns the merge code entered in the **Merge code** field of the **Merge Forms** dialog box.
&PARAM9 (history record)	Returns the RecNo or RecID of the history record created by GoldMine. This macro is useful for updating the history record.

Macros for the GoldMine License

The following macros return data for the current GoldMine license. The descriptions for each macro include the corresponding field name from the form that appears in the Registration tab of the *GoldMine Net-Update* window. For details on the Net-Update process, see “Using Net-Update” in the online Help.

&LicInfoLicTo	Returns the *Organization* entry from the registration form.
&LicInfo_Contact	Returns the *Contact Name* entry from the registration form.
&LicInfo_LicEmail	Returns the *E-mail address* entry from the registration form.

&LicInfo_Phone	Returns the telephone number entry from the first *Phone/Fax* field.
&LicInfo_Fax	Returns the fax number entry from the second *Phone/Fax* field.
&LicInfo_Address1	Returns the *Address1* entry from the registration form.
&LicInfo_Address2	Returns the *Address2* entry from the registration form.
&LicInfo_City	Returns the city entry from the first *City/State* field.
&LicInfo_State	Returns the state or province entry from the second *City/State* field.
&LicInfo_Zip	Returns the ZIP Code entry from the first *Zip/Country* field.
&LicInfo_Country	Returns the country entry from the second *Zip/Country* field.

Controlling the GoldMine User Interface

There are a number of commands that allow the programmatic control of the GoldMine user interface. For example, menu commands can be executed; controls can be populated, enabled, or disabled; and windows can be allowed to launch or vetoed.

There are three general groups of commands to accomplish these tasks. The first group of commands provides information as to the windows and dialogs available to be controlled and the methods to subscribe to events concerning those windows. The second group of commands manipulates the controls on GoldMine's windows and dialog boxes. The final group is event methods that are implemented in the intregration to handle events that are raised based on the events subscribed to.

NOTE: The events in the GoldMine.UI class require a command to be called to subscribe to the desired event. The events in the GoldMine.RecObj class and the GoldMine.GMSystemEvents class do not require subscription.

Getting Window Information

The GetAvailableWindowsList and GetActiveWindowsList commands return information about the available and active windows in GoldMine. This information is needed to supply data to the event subscription commands and control manipulation commands.

GetAvailableWindowsList

GetAvailableWindowsList returns all of the available GoldMine windows in XML format.

Syntax

XML	`<GMAPI call="GetAvailableWindowsList"/>`

Returned XML

The XML returned is a long list of available windows for GoldMine. It has the following format. This represents a truncated list of available windows. The actual list is too extensive to list in this document. All window names are descriptive and self-explanatory as to which window they represent. Send the GetAvailableWindowList command for a complete list of windows.

```
<GMAPI call="GetAvailableWindowsList">
<status code="1">Success</status>
<data name="WindowsList">
<data name="window">DIALOGFILEDFOLDERPROPERTIES</data>
<data name="window">DIALOGMAILSEARCH</data>
<data name="window">DIALOGEMAILACCNTPROPS</data>
<data name="window">DIALOGEMAILAUTOFILEMONTH</data>
<data name="window">DIALOGDIGITALIDEXPORTPRIVATE</data>
<data name="window">DIALOGSOFTPHONE</data>
<data name="window">DIALOGSIP_SP_SETTINGS</data>
</data>
</GMAPI>
```

GetActiveWindowsList

The GetActiveWindowsList supplies detailed information regarding the windows and dialog boxes currently active in GoldMine.

Syntax

XML	`<GMAPI call="GetActiveWindowsList"/>`

Returned XML

Below is an example XML document describing one active window, the current contact screen. For an accurate representation of the window you wish to control, call GetActiveWindowsList with that window active. Doing so will provide a reference for programming your integration.

All window elements are stored in the WindowsList element. Each Window has child elements providing detailed information about the window. Some child elements store additional child elements when further nesting is required to provide all properties of the windows and the controls they contain. Commands that manipulate the controls on a window expect the handle the parent window (hwnd) and the control's id, along with the properties of the control that are being changed. Retrieve the hwnd and the control id from the GetActiveWindowsList command.

```
<GMAPI call="GetActiveWindowsList">
<status code="1">Success</status>
<data name="WindowsList">
<data name="window">
<data name="hWnd">197868</data>
<data name="WindowName">OBJECTCURRENTGMRECORD</data>
<data name="WindowInternalName">OBJECT: GMRECORD</data>
<data name="Caption">GoldMine, Inc.</data>
<data name="WinType">Window</data>
<data name="WindowRect">
<data name="Left">140</data>
```

```
<data name="Right">722</data>
<data name="Bottom">484</data>
<data name="Top">81</data>
</data>
<data name="ClientRect">
<data name="Left">144</data>
<data name="Right">718</data>
<data name="Bottom">480</data>
<data name="Top">111</data>
</data>
<data name="Controls">
<data name="msctls_updown32">
<data name="Enabled">1</data>
<data name="Visible">1</data>
<data name="ParentID">197868</data>
<data name="hWnd">1770672</data>
<data name="ID">700</data>
</data>
<data name="msctls_updown32">
<data name="Enabled">1</data>
<data name="Visible">1</data>
<data name="ParentID">197868</data>
<data name="hWnd">66798</data>
<data name="ID">704</data>
</data>
<data name="gmWndBrowse">
<data name="Enabled">1</data>
<data name="Visible">1</data>
<data name="ParentID">197868</data>
<data name="hWnd">66812</data>
<data name="ID">1003</data>
<data name="Text">History of GoldMine, Inc.</data>
<data name="Controls">
<data name="ScrollBar">
<data name="Enabled">1</data>
<data name="Visible">1</data>
<data name="ParentID">66812</data>
<data name="hWnd">66814</data>
<data name="ID">100</data>
</data>
</data>
</data>
</data>
</data>
</GMAPI>
```


Registering for Events

Before you can receive events from the GoldMine.UI class, you need to subscribe to the specific events you wish to receive for the desired windows.

NOTE: When using Visual Basic 6.0, be sure to declare your GoldMine objects using the WithEvents qualifier.

NOTE: Dim WithEvents GMObj as GoldMine.UI

RegisterVetoWindowLaunch

RegisterVetoWindowLaunch subscribes to an event for the specified window giving the integration the opportunity to either veto or allow the window launch.

Syntax

XML	`<GMAPI call="RegisterVetoWindowLaunch" > <data name=”Window”> DIALOGSCHEDULEDEFAULT</data> <data name=”Monitor”>1</data> </GMAPI>`

Parameters

Window: the name of the window to monitor. The GetAvailableWindowsList command provides valid window names.

NOTE: Only dialog boxes can be vetoed. For example, the schedule and complete windows are dialog boxes. Core GoldMine windows cannot be vetoed (the record object, the email center, etc)

Monitor: specifies to either begin monitoring for the event (1) or to unsubscribe from the event (0).

Returned XML

The following XML is returned:

```
<GMAPI call="RegisterVetoWindowLaunch">
<status code="1">Success</status>
</GMAPI>
```

For information on handling the event, see .

RegisterWindowUpDown

RegisterWindowUpDown subscribes to an event for the specified window notifying the integration when the desired window is launching or closing.

Syntax

XML	`<GMAPI call="RegisterWindowUpDown" > <data name=”Window”> DIALOGSCHEDULEDEFAULT</data> <data name=”Monitor”>1</data> </GMAPI>`

Parameters

Window: the name of the window to monitor. The GetAvailableWindowsList command provides valid window names.

Monitor: specifies to either begin monitoring for the event (1) or to unsubscribe from the event (0).

REturned XML

The following XML is returned:

```
<GMAPI call="RegisterWindowUpDown">
<status code="1">Success</status>
</GMAPI>
```

For information on handling the event, see .

RegisterCommandExec

RegisterCommandExec is used to subscribe to events raised when a particular control is manipulated on the specified window. For example, your application can receive notification when the user combo (dropdown) box is changed on the Schedule a Call dialog.

Syntax

XML	`<GMAPI call="RegisterCommandExec">`
	`<data name="Window">DialogScheduleDefault</data>`
	`<data name="ControlID">1</data>`
	`<data name="CommandID">0</data>`
	`<data name="Monitor">1</data>`
	`</GMAPI>`

Parameters

Window: The name of the window to monitor. The GetAvailableWindowsList command provides valid window names.

ControlID: The ID of the control to monitor. This ID is provided in the child elements for the specified window provided by the GetAvailableWindowsList.

CommandID: The type of event to monitor (i.e. button clicked). The possible values for the CommandID are enumerated within the GoldMine object. Provided notification command ID's include ButtonStates, ComboBoxStates, EditControlNotifications, and ListBoxNotifications.

NOTE: The CommandID enumerations can be viewed in the Object Browser in Visual Basic 6.0

Monitor: Specifies to either begin monitoring for the event (1) or to unsubscribe from the event (0).

Returned XML

The following XML is returned:

```
<GMAPI call="RegisterCommandExec">
<status code="1">Success</status>
</GMAPI>
```

For information on handling the event, see .

RegisterTabDetailsEvent

RegisterTabDetailsEvents is used to subscribe to events raised when a particular Record Object Tab is manipulated. For example, your application can receive notification when the user clicks on an item in a tab, but without the item being zoomed or opened.

Syntax

XML	`<GMAPI call="RegisterTabDetailsEvents">`
	`<data name="Monitor">1</data>`
	`</GMAPI>`

Parameters

Monitor: Specifies to either begin monitoring for the event (1) or to unsubscribe from the event (0).

The following tab events are monitored:

Event	Data Passed
AdditionalContactClick	RecID,AccountNo,Reference,Phone,Contact
AdditionalContactEditClick (7.5 or higher)	RecID,AccountNo,Reference,Phone,Contact
AdditionalContactNewClick (7.5 or higher)	AccountNo (of the contact it will be attached to)
DetailsClick	RecID,AccountNo,Type,Reference
DetailsEditClick (7.5 or higher)	RecID,AccountNo,Type,Reference
DetailsNewClick (7.5 or higher)	AccountNo
ReferralClick	RecID,LinkedRecID,LinkedAccountNo,Referral,Reference
ReferralAddClick	RecID (the recid of the referrer,not the referree)
ReferralEditClick (7.5 or higher)	RecID,LinkedRecID,LinkedAccountNo,Referral,Reference
LinkedDocClick	RecID,FileName,Sync,UserName
LinkedDocAddClick	Returns Account No of current contact
LinkedDocEditClick (7.5 or higher)	RecID,FileName,Sync,UserName
PendingEditClick (7.5 or higher)	RecID,AccountNo,RecType,UserName
PendingClick	RecID,AccountNo,RecType,UserName
ScheduleNew (7.5 or higher)	AccountNo,RecType,UserName
HistoryEditClick (7.5 or higher)	RecID,AccountNo,RecType,UserName
HistoryClick	RecID,AccountNo,RecType,UserName

The following Case tab events are also monitored. Each event returns the RecID of the selected case:

Event (All are 8.0 or higher only)	User Action	Returns
CaseReassign	Reassign the case	RecID
CaseEscalate	Escalate the case	RecID
CaseResolve	Resolve the case	RecID
CaseAbandon	Abandon the case	RecID
CaseGoto	Open the case	RecID
CaseSaveAsTemplate	Save the case as a template	RecID
CaseDelete	Delete the case	RecID

AdditionalContactClick

AdditionalContactClick

Returned XML

The following XML is returned for AdditionalContactClick:

```
<GMAPI event="AdditionalContactClick">
<RecID>99UZA3O%R*O%H?$</RecID>
<AccountNo>A1121345737(&gt;C9^HBob</AccountNo>
<Reference/>
<Phone/>
<Contact>Frances</Contact>
</GMAPI>
```

Parameters

RecID: The record ID for the additional contact.

AccountNo: The account number of the parent contact.

Reference: The reference field value.

Phone: The phone field value.

DetailsClick

Returned XML

The following XML is returned for DetailsClick:

```
<GMAPI event="DetailsClick">
<RecID>99UZC5R(*2!2H?$</RecID>
<AccountNo>A1121345737(&gt;C9^HBob</AccountNo>
<Type>E-mail Address</Type>
<Reference>some.email@domain.com</Reference>
</GMAPI>
```

Parameters

RecID: The record ID for the detail.

AccountNo: The account number of the contact.

Type: The type of the detail.

Reference: The reference field value.

PendingClick

Returned XML

The following XML is returned for PendingClick:

```
<GMAPI event="PendingClick">
<RecID>BA5OXQT%ZO9K]WV</RecID>
<AccountNo>A1121345737(&gt;C9^HBob</AccountNo>
<RecType>C</RecType>
<UserName>GUY</UserName>
</GMAPI>
```

Parameters

RecID: The record ID for the pending item.

AccountNo: The account number of the contact.

RecType: The record type of the pending item.

UserName: The owner name.

HistoryClick

Returned XML

The following XML is returned for HistoryClick:

```
<GMAPI event="HistoryClick">
<RecID>BA4U3BK%BK!J]WV</RecID>
<AccountNo>A1121345737(&gt;C9^HBob</AccountNo>
<RecType>L</RecType>
<UserName>GUY</UserName>
</GMAPI>
```

Parameters

RecID: The record ID for the history item.

AccountNo: The account number of the contact.

RecType: The record type of the history item.

UserName: The owner name.

LinkedDocClick

Returned XML

The following XML is returned for LinkedDocClick:

```
<GMAPI event="LinkedDocClick">
<RecID>BAAVH43(C?LC]WV</RecID>
<FileName>C:\documents and settings\john stillman\my documents\visual
studio projects\gmdev\bin\debug\MailBox\Attach\There ya go2.doc</FileName>
<Sync>1</Sync>
<UserName>GUY</UserName>
</GMAPI>
```

Parameters

RecID: The record ID for the linked document.

FileName: The path to the linked document.

Sync: 1 or 0 for is the doc synced.

UserName: The last user to use the document (not the owner).

For information on handling these events, see .

Handling GoldMine.UI Events

There are four events in the GoldMine.UI class that can be utilized. In order to be notified of the events, the integrating application must register with GoldMine via the above commands.

This section will show examples of handling these events in VB and VB.NET. The method to handle the events may vary depending on the development environment being used.

NotifyControlCommand

NotifyControlCommand is the event that notifies a client application that a button has been pressed, a checkbox marked, or any other control change/activation event. Register for this event by calling *RegisterCommandExec*.

Parameters

sWindowName: This is a string (BSTR) that contains the nam of the window being called.

ControlID: a long that contains the ID of the control that is notifying.

CmdID: a long that contains the command that is being triggered

HWnd: a long that represents the hWnd of the Parent to the control.

VetoWindow

The VetoWindow event is used to notify a client application that a window or dialog is requesting to be launched. The client application returns a Boolean answer as to whether or not to allow the window/dialog to launch. Subscribe to this event by calling RegisterVetoWindowLaunch.

Parameters

sWindowName: a string (BSTR) that contains the name of the window being called.

NOTE: Delphi does not support functions (a sub that returns a value) in its COM handler. Within the VetoWindow event handler, Delphi users need to set a special property within the GoldMine.UI class to indicate whether or not to veto the window. For Example: *GMObj.VetoWindowDelphi:=true*

Example

The following example uses Visual Basic 6.0. After declaring your object using the WithEvents keyword, Visual Basic will place the name of the object in the drop down on the upper left of your code window. Select your object from that drop down to view the list of event handling subs/functions available for that object. For the VetoWindow event the function will be called Objectname_VetoWindow. For an example handling an event in VB.NET using delegate functions, see the GoldMineShutdown event for the GoldMine.GMSystemEvents class.

```
Private Function GMObj_VetoWindow(ByVal sWindowName As String) As Boolean
If sWindowName = "DIALOGSCHEDULEDEFAULT" Then
Dim sResult As String
Dim iRes As Integer

sResult = GMObj.ExecuteCommand("<GMAPI call=""MsgBox""> 
<data name=""Message"">Do you want to bring up the GoldMine schedule
window?
</data><data name=""Style"">4</data></GMAPI>")

Dim docResult As DOMDocument40
Set docResult = New DOMDocument40

docResult.loadXML sResult

Dim elRoot As IXMLDOMElement
Set elRoot = docResult.documentElement
Dim att As IXMLDOMNode
Set att = elRoot.childNodes(0)
If att.Attributes(0).baseName = "code" Then
iRes = att.Text
End If
If iRes = 6 Then
GMObj_VetoWindow = False
Else
GMObj_VetoWindow = True
End If
Set docResult = Nothing
Set elRoot = Nothing
Set att = Nothing

End If

End Function
```

WindowUpDown

The purpose of the WindowUpDown event is to notify the client application that a particular window is coming up or shutting down. This does not apply to the main GoldMine application window. To be notified that GoldMine is shutting down, use the GoldMineShutdown event in the GoldMine.GMSystemEvents class.

This event is useful for a client application to perform additional processing of record data after the user has submitted it by pressing OK on a dialog box. For example, data can be linked to other third party applications in real time.

Parameters

sName: a string (BSTR) that contains the name of the window being called.

bUp: a Boolean which represents True=Up and False=Down

GMEvent

GMEvent is an omni-event holder that can provide information about what is happening in the GoldMine application, and in some cases it can affect an action in GoldMine.

```
VARIANT_BOOL GMEvent(VARIANT_BSTR sXML)
```

sXML is XML that describes the event - possible events are UI events:

VetoWindow - same as the 6.7 event - looks like

```
<GMAPI event="VetoWindow">
<WindowName>NAME_OF_WINDOW_HERE</WindowName>
</GMAPI>
```

If event returns TRUE to GM then the window will not be launched

WindowUpDown - same as the 6.7 event - returns

```
<GMAPI event="WindowUpDown">
<WindowName>NAME_OF_WINDOW_HERE</WindowName>
<Up/>
<WindowhWnd>399692</WindowhWnd>
</GMAPI>
```

If the window is being closed, then a Down node will appear instead of the Up node

NotifyControlCommand - same as the 6.7 event - returns

```
<GMAPI event="NAME_OF_WINDOW_HERE">
<WindowName>DIALOGSCHEDULEDEFAULT</WindowName>
<ID>1</ID>
<Command>0</Command>
<WindowhWnd>97256300</WindowhWnd>
</GMAPI>
```

The following are the new events specific to 7.0 and only can be used with the GMEvent structure

CalendarMonthView_DaySelectedWithActivities - event to show when a user has clicked a day with activities in the month view.

Returns

```
<GMAPI event="CalendarMonthView_DaySelectedWithActivities">
<Date>20150624</Date>
<Timed>0</Timed>
<Timeless>1</Timeless>
<Events>0</Events>
</GMAPI>
```

Date - is the date clicled in YYYYMMDD format

Timed - the number of timed activities on that day

Timeless - the number of timeless activities

Events - the number of events on that day

CalendarDayActivityHighlighted - for week and day views, shows the details of an activity that a user has clicked on

```
<GMAPI event="CalendarDayActivityHighlighted">
<ActvAccNo>A4032327210$Z7/!R </ActvAccNo>
<CalRecID>B6AANW4#Y&gt;N(]WV</CalRecID>
<Contact>Dan Gorentz</Contact>
<CreatedBy>GUY </CreatedBy>
<User>GUY </User>
</GMAPI>
```

ActvAccNo - the contact AccountNo that this cal entry belongs to

CalRecID the record id of the calendar entry

Contact - the contact field for the record

CreatedBy - the user that created the record

User - the user its assigned to

VetoCalendarChangeView - can block the view from changing tabs

```
<GMAPI event="VetoCalendarChangeView">
<PrvView>1</PrvView>
<NewView>2</NewView>
</GMAPI>
```

View are enumerated as follows

0 - Day View

1 - Week View

2 - Month

3 - Year

4 - Planner

5 - Outline

6 - PegBoard

PrvView - the view it is changing from

NewView - the view it is changing to

Returning TRUE to this event blocks the view change

CalendarUserSelectionChanged - tells the consumer that the user selection of visible user events has changed.

```
<GMAPI event="CalendarUserSelectionChanged">
<Users>GUY,MASTER</Users>
<CurrentView>0</CurrentView>
</GMAPI>
```

Users - a comma delimited list of users that are shown in the calendar.

CurrentView - the current view

VetoCalendarNextClick - can block the user from hitting the next button

Returns

```
<GMAPI event="VetoCalendarNextClick"/>
```

returning TRUE to this event keeps the user on the current selection

VetoCalendarPreviousClick - can block the user from hitting the previous button

```
<GMAPI event="VetoCalendarPreviousClick"/>
```

returning TRUE to this event keeps the user on the current selection

Manipulating Controls Programatically

The GoldMine.UI class responds to various commands to programmatically manipulate the controls on GoldMine's dialog boxes.

To specify the control to change or activate, read the parent window's handle (hwnd) and the control's ID from the GetActiveWindowsList command. The control ID's will always stay the same and will be unique only to the scope of the dialog they exist on. In other words, the GoldMine user drop down box on the Schedule a Call dialog will always have the same control ID. This control ID can be discovered during the design phase of your application. Use the control ID as the identifier for checking the state of the control when reading the control properties from the GetActiveWindowsList command.

PressButton

Use PressButton to press a button on a known form.

Syntax

XML

GetActiveWindowsList returned a window with the following control:

```
<data name="Button">
<data name="Enabled">1</data>
<data name="Visible">1</data>
<data name="ParentID">2232874</data>
<data name="hWnd">987600</data>
<data name="ID">2060</data>
<data name="Text">&amp;Activate</data>
</data>
```

To press this button, the following XML should be sent:

```
<GMAPI call="PressButton">
<data name="hWndParent">2232874</data>
<data name="ID">2060</data>
</GMAPI>
```

NOTE: The hWndParent parameter of the PressButton command corresponds to the ParentID returned for the control from GetActiveWindowsList, not hWnd, which is the hWnd of the control. Also, the ID parameter corresponds to the ID parameter of the control returned by the GetActiveWindowsList, not the hWnd.

Parameters

hWndParent: the handle to the parent window containing the control. Corresponds to the ParentID element returned for the control by the GetActiveWindowsList command.

ID: the ID of the control. Corresponds to the ID element returned for the control by the GetActiveWindowsList command.

SetControlText

SetControlText sets the text property of the specified control.

Syntax

XML

The Filters and Groups dialog contains the following control, the SQL field:

```
<data name="Edit">
<data name="Enabled">1</data>
<data name="Visible">1</data>
<data name="ParentID">398370</data>
<data name="hWnd">726100</data>
<data name="ID">104</data>
</data>
```

To set the text for this control, the following XML should be sent:

```
<GMAPI call="SetControlText">
<data name="hWndParent">398370</data>
<data name="ID">104</data>
<data name="Text">SELECT * FROM contact1</data>
</GMAPI>
```

Parameters

hWndParent: the handle to the parent window containing the control. Corresponds to the ParentID element returned for the control by the GetActiveWindowsList command.

ID: the ID of the control. Corresponds to the ID element returned for the control by the GetActiveWindowsList command.

Text: the text desired for the control.

SetCheckBox

SetCheckBox sets the value of a check box control.

Syntax

XML

A dialog has the following control:

```
<data name="Button">
<data name="Enabled">1</data>
<data name="Visible">1</data>
<data name="ParentID">199202</data>
<data name="hWnd">199212</data>
<data name="ID">111</data>
<data name="Text">&amp;Master rights</data>
</data>
```

To set the checkbox, the following XML should be sent:

```
<GMAPI call="SetCheckBox">
<data name="hWndParent">199202</data>
<data name="ID">111</data>
<data name="Checked">1</data>
</GMAPI>
```

Parameters

hWndParent: the handle to the parent window containing the control. Corresponds to the ParentID element returned for the control by the GetActiveWindowsList command.

ID: the ID of the control. Corresponds to the ID element returned for the control by the GetActiveWindowsList command.

Checked: 1 to check the checkbox, 0 to uncheck

SelectRadio

The SelectRadio command allows an application to set a radio button array, or a single item. While the command allows a single radio button to be set, this is not the best practice. Doing so results in more than one radio button selected in a group or radio buttons.

Syntax

XML

A dialog has the following two controls:

```
<data name="Button">
<data name="Enabled">1</data>
<data name="Visible">1</data>
<data name="ParentID">330708</data>
<data name="hWnd">134108</data>
<data name="ID">532</data>
<data name="Text">&amp;Dark Background</data>
</data>
<data name="Button">
<data name="Enabled">1</data>
<data name="Visible">1</data>
<data name="ParentID">330708</data>
<data name="hWnd">134106</data>
<data name="ID">533</data>
<data name="Text">&amp;Bright Background</data>
</data>
```

To select the Dark Background radio and unselect the Bright Background, the following XML should be sent:

```
<GMAPI call="SelectRadio">
<data name="RadioButton">
<data name="hWndParent">199516</data>
<data name="ID">532</data>
<data name="Value">1</data>
</data>
<data name="RadioButton">
<data name="hWndParent">199516</data>
<data name="ID">533</data>
<data name="Value">0</data>
</data>
</GMAPI>
```

Parameters

hWndParent: the handle to the parent window containing the control. Corresponds to the ParentID element returned for the control by the GetActiveWindowsList command.

ID: the ID of the control. Corresponds to the ID element returned for the control by the GetActiveWindowsList command.

Value: 1 to select the radio button, 0 to unselect

SetListBox/SetComboBox

Use the SetListBox/SetComboBox command(s) to select an item in a listbox on a GoldMine dialog box. The client application can specify either a text value or an index. If a text value is used, the value must already exist within the list.

Syntax

XML

A dialog has the following control:

```
<data name="ComboBox">
<data name="Enabled">1</data>
<data name="Visible">1</data>
<data name="ParentID">330654</data>
<data name="hWnd">68972</data>
<data name="ID">537</data>
<data name="Text">MMM d, yy </data>
</data>
```

To select a different item in this combobox, use the following XML:

Using an Index:

```
<GMAPI call="SetComboBox">
<data name="hWndParent">330654</data>
<data name="ID">537</data>
<data name="Index">0</data>
</GMAPI>
```

Using a Text value:

```
<GMAPI call="SetComboBox">
<data name="hWndParent"> 330654</data>
<data name="ID">537</data>
<data name="Value">MMMM dd, yyyy</data>
</GMAPI>
```

NOTE: SetComboBox and SetListBox have been grouped together in this document because they share the same parameters and functionality for their respective control. However, SetComboBox should only be used for comboboxes and SetListBox for listboxes.

Parameters

hWndParent: the handle to the parent window containing the control. Corresponds to the ParentID element returned for the control by the GetActiveWindowsList command.

ID: the ID of the control. Corresponds to the ID element returned for the control by the GetActiveWindowsList command.

Value: the TEXT value to select in the combobox or listbox. The value must already exist in the list of the control.

OR

Index: the index number of the item to be selected in the combo box or list box.

SelectTab

Use SelectTab to select a particular tab on a dialog box. This command does not select the tabs on the contact record. Use the SetRoTabX command for that purpose.

Syntax

XML	A dialog has the following control:
	`<data name="SysTabControl32">`
	`<data name="Enabled">1</data>`
	`<data name="Visible">1</data>`
	`<data name="ParentID">789580</data>`
	`<data name="hWnd">330824</data>`
	`<data name="ID">12320</data>`
	`</data>`
	
	To select the tab with index of 1:
	`<GMAPI call="SelectTab">`
	`<data name="hWndParent">789580</data>`
	`<data name="ID">12320</data>`
	`<data name="Index">1</data>`
	`</GMAPI>`

NOTE: The SelectTab command may not function as expected on all tabs within GoldMine. Due to the way some dialog boxes were developed, changing the tab with the SelectTab command may not cause the correct controls to be displayed on the desired tab. Always test the SelectTab command on the dialog box you wish to execute it for during development of your application to verify it correctly switches the tab.

Parameters

hWndParent: the handle to the parent window containing the control. Corresponds to the ParentID element returned for the control by the GetActiveWindowsList command.

ID: the ID of the control. Corresponds to the ID element returned for the control by the GetActiveWindowsList command.

Index: the index number of the tab to be selected.

EnableCtrl

The EnableCtrl command allows the programmer to enable or disable any control.

Syntax

XML

A dialog has the following control:

```
<data name="Button">
<data name="Enabled">1</data>
<data name="Visible">1</data>
<data name="ParentID">789580</data>
<data name="hWnd">1117262</data>
<data name="ID">1</data>
<data name="Text">OK</data>
</data>
```

To disable the button:

```
<GMAPI call="EnableCtrl">
<data name="hWndParent"> 789580</data>
<data name="ID">1</data>\
<data name="Enable">0</data>
</GMAPI>
```

Parameters

hWndParent: the handle to the parent window containing the control. Corresponds to the ParentID element returned for the control by the GetActiveWindowsList command.

ID: the ID of the control. Corresponds to the ID element returned for the control by the GetActiveWindowsList command.

Enable: set to 1 to enable the control, 0 to disable.

Executing a Menu Command

The MenuCommand function allows the programmatic execution of a menu item, as if the user has clicked the item in the GoldMine menu.

Syntax

XML

```
<GMAPI call="MenuCommand" >FileNewRecord</GMAPI>
```

OR

```
<GMAPI call="MenuCommand">
<data name="MenuCommand">FileNewRecord</data>
</GMAPI>
```

MenuCommand accepts one parameter, MenuCommand. This parameter can be any of the following menu commands. The command name is descriptive and indicates which menu item it corresponds to:

FileNewRecord	FileNewRecordToExistingCompany	FileNewRecordAndOrgChart
FileNewRecordToExistingOrgChart	FileNewRecordByType	FileOpenDatabase
FilePrint1Report	FileNewDatabase	FileMaintainDatabases

FileBackupDatabases	FileRestoreDatabases	FilePrintReports
FileSetupPrinter	SynchronizationOneButtonSync	SynchronizationWizard
GoldSyncAdministrationCenter	SynchronizeWithOutlook	SynchronizeWithPilot
SynchronizeWithWindowsCEPDA	FileCopyMoveRecords	ConfigureUsersSettings
ConfigureUserGroups	ConfigureResources	ConfigureRecordType
ConfigureCustomScreens	ConfigureCustomFields	ConfigureHTMLTab
ConfigureSyncSettings	ConfigureLicenseManager	ConfigureMyGoldMine
LogAway	LogInAnotherUser	LogInServiceSupport
Exit	EditUndo	EditCut
EditCopy	EditPaste	EditCopyContactDetails
EditContact	DeleteContact	Record-related Settings
Contact Details	RecordDetailsOrganization	RecordDetailsSummary
RecordDetailsFields	RecordDetailsHTMLTab	RecordDetailsNotes
RecordDetailsContacts	RecordDetailsDetails	RecordDetailsReferrals
RecordDetailsPending	RecordDetailsHistory	RecordDetailsLinks
RecordDetailsMembers	RecordDetailsTracks	RecordDetailsOpptys
RecordDetailsProjects	RecordDetailsTickets	RecordDetailsResize
TimerStart	TimerStop	TimerReset
TimerRestart	EditToolbars	EditCustomTemplates
EditPreferences	ViewMyGoldMine	ViewNewContactWindow
ViewContactGroups	ViewCalendar	ViewActivityList
ViewEmailCenter	ViewEmailWaitingOnline	ViewInfoCenter
ViewProjects	ViewPersonalRolodex	ViewLiteratureFulfillment
SalesToolsOpportunities	SalesToolsScripts	AnalysisSales
AnalysisStatistical	AnalysisForecast	AnalysisGraphical
AnalysisLeads	AnalysisQuota	ViewGoldMineLogs
ViewSyncRetrievalLogs	LookupCompany	LookupContact
LookupLastName	LookupPhone	LookupZIPCode

LookupCity	LookupState	LookupCountry
LoookupAccountNo	LookupKey1	LookupKey2
LookupKey3	LookupKey4	LookupKey5
LookupDetailRecords	LookupEmailAddress	LookupAdditionalContName
LookupFilters	LookupSQLQueries	TextSearchPrimaryFields
TextSearchNotes	TextSearchAllFields	TextSearchFieldsBelowTabs
GotoNextRecord	GotoPreviousRecord	GotoCycleLastViewedRecords
GotoLastRecord	GotoRecordNumber	GotoFirstRecord
DialPhone1	DialPhone2	GotoInternetSearch
DialFax	RedialLastNumber	DialPhone3
IncomingCall	ContactInsertNote	ManualDial
WriteMemoToContact	WriteFAXtoContact	WriteLetterToContact
ContactWriteCustomizeTemplates	WriteCustomizeTemplates	WriteMailMerge
EmailOutlookMessageToContact	EmailPagerMessageToContact	EmalMessageToContact
EmailCustomizeTemplates	ContactTakePhoneMessage	EmailMerge
ContactBrowseWebStie	LinkFile	ContactAssignProcess
ScheduleCall	ScheduleNextAction	AddDetail
ScheduleLiteratureRequest	ScheduleForecastedSale	ScheduleAppointment
ScheduleEvent	ScheduleTodo	ScheduleOtherAction
CompleteScheduledCall	CompleteUnscheduledOutgoingCall	ScheduleGoldMineEmail
CompleteMessage	CompleteNextAction	CompleteUnscheduledIncomingCall
CompleteSale	CompleteOtherAction	CompleteAppointment
CompleteToDo	CompleteLetterMemo	CompleteEvent
CompletePendingActivities	AutomatedProcessesExecute	CompleteLiteratureRequest
AutomatedProcessesSetup	ServerAgenstStart	AutomatedProcessesRemoveTrack
ActImport	OutlookImport	ServerAgentsAdministrator
ExportContactRecords	ImportZIPCodes	ImportContactRecords
XMLImport	XMLExport	RunQSW

ICALExport	CalPublish	ICALImport
ToolsCleanupDOSNotes	ToolsOptimizeOrgChartAccess	PublishBusyTime
ToolsTerritoryRealignment	MergePurgeWizard	ToolsGlobalReplaceWizard
MergeTaggedRecords	ToolsDeleteRecordsWizard	MergeVisibleRecords
ToolsStrategicSolutions	ToolsBDEAdministrator	ToolsSyncSpy
WindowTile	WindowTileWide	ToolsSystemPerformance
WindowArrangeIcons	WindowCloseAll	WindowCascade
WindowStatusBar	WindowTaskBar	WindowToolBar
HelpHelpTopics	HelpReleaseNotes	WindowBackgroundSettings
HelpNewsgroups	HelpUpdateGoldMine	HelpGoldMineWebSite
CampaignManager	LeadCenter	HelpAbout
WebImportAdmin		

Returned XML

The MenuCommand function returns after the menu command is executed. It does not wait for any events on the resulting window before returning. The returned XML for a successful call will be:

```
<GMAPI call="MenuCommand"><status code="1">The command was
executed.</status></GMAPI>
```

In the event that there is a modal window active in the GoldMine user-interface, the COM Server cannot launch another window (as would be the case if attempting to launch a menu item within the interface). When that occurs, the following XML is returned to indicate a failure:

```
<GMAPI call="MenuCommand">
<status code="0">Access is denied.</status>
</GMAPI>
```

Opening a Mail Record

The OpenMailRecord function opens a mail record in the mail center when the RecID of the mail item is passed.

Syntax

XML	To open a mail record:
	`<GMAPI call=”OpenMailRecord”>`
	`<data name=”RecID”> 789580</data>`
	`</GMAPI>`

Parameters

RecID: the record ID of the mail item.

Returned XML

The OpenMailRecord function returns after the command is executed. The returned XML for a successful call will be:

```
<GMAPI call="OpenMailRecord"><status code="1">The command was
executed.</status></GMAPI>
```

In the event that the mail record is already open, the following XML is returned to indicate a failure:

```
<GMAPI call="OpenMailRecord">
<status code="-1">Already open.</status>
</GMAPI>
```

In the event that the system cannot open the mail record, the following XML is returned to indicate a failure:

```
<GMAPI call="OpenMailRecord">
<status code="0">Failure.</status>
</GMAPI>
```

Setting a Selected Record in a GoldMine Grid (GoldMine 8.0 or higher)

SetGridRecID allows you to set the selected record in a given GoldMine grid.

In the following example, you can set the Linked Document tab to a certain row:

1. We call SetROTab with a value of 10 to set the Link tab to focus
2. Perform GetActiveWindowList
3. Look for the gmWndBrowse object to retain it's hWnd value.
4. Call the SetGridRecID function (see example)
5. If you had registered for Tab events, then you would also get the event

```
<GMAPI event="LinkedDocClick">
<RecID>CHNHXID(2AAS]WV</RecID>
<FileName></FileName>
<Sync>1</Sync>
<UserName>GUY</UserName>
</GMAPI>
```

Syntax (Example)

XML	To set a selected record in a grid:
	`<GMAPI call="SetGridRecID">`
	`<data name="hWnd">1057444</data>`
	`<data name="RECID">CHNHXID(2AAS]WV</data>`
	`</GMAPI>`

Parameters

hWnd: The hWnd of the gmWndBrowse you wish to set.

RecID: The recid of the value in the list you wish to select. You must pass a valid recid that is represented in the grid.

Returned XML

The returned XML for a successful call will be:

```
<GMAPI call="SetGridRecID">
```

```
<status code="1">Success</status>
</GMAPI>
```

Returning Selected Records in a GoldMine Grid (8.0.1 or higher)

GetGridRecID returns the selected records in a given GoldMine grid.

Syntax (Example)

XML	To get selected records in a grid:
	`<GMAPI call="GetGridRecID">`
	`<data name="HWND">337700</data>`
	`</GMAPI>`
	or
	`<GMAPI call="GetGridRecID">468730</GMAPI>`

Parameters

hWnd: The hWnd of the gmWndBrowse from which you wish to get selected recids.

Returned XML

The returned XML for a successful call will be:

```
GMAPI call="GetGridRecID">
<status code="1">Success</status>
<data name="Return">
<data name="RecID">CGNPHUE)D0TV W&lt;</data>
</data>
</GMAPI>
```

Or if there are multiple items selected:

```
<GMAPI call="GetGridRecID">
<status code="1">Success</status>
<data name="Return">
<data name="RecID">AO6R9GO$/X^1$M&lt;</data>
<data name="RecID">ANWYLNL%XV]&amp; W&lt;</data>
<data name="RecID">AOCJ5LF)&gt;ED0 W&lt;</data>
<data name="RecID">AOCJ5LF+Y-(8 W&lt;</data>
<data name="RecID">AOCJ5PO#E,5/ W&lt;</data>
<data name="RecID">AWUX7WW :U3Z W&lt;</data>
</data>
</GMAPI>
```

GoldMine.RecObj Class

The GoldMine.RecObj class contains only events. These events notify the client application when the record object has changed, when a field has changed on the contact record, or when the tab selected on the record object has changed. It is not necessary to subscribe to these events, just implement the event handlers.

RecordObjectHasChanged

The RecordObjectHasChanged event indicates when the contact displayed in GoldMine has changed to a different contact. This does not indicate data changes. This event is the equivalent of setting the LinkMode in Visual Basic to vbLinkNotify.

Parameters

sCurrentRecord: a string that contains the AccountNo of the current record.

RecordFieldHasUpdated

The RecordFieldHasUpdated event indicates when the value of a field in contact1 or contact2 for the current contact has been updated. This event does NOT notify when an Email Address or Web Site has changed.

Parameters

sField: a string that contains the fieldname of the updated field.

sLabel: the local label (or global if no local label is specified) of the field.

ContactTableID: the ID number of the contact table. Will be 1 for contact1 and 2 for contact2.

RecordTabHasChanged

The RecordTabHasChanged event indicates when the user in GoldMine has selected a different tab at the bottom of the contact record screen.

Parameters

sCurrentTab: the numeric representation of the tab selected.

GoldMine.GMSystemEvents Class

The GoldMine.GMSystemEvents class contains one event, GoldMineShutDown, indicating when the GoldMine application is shutting down. This gives the client application an opportunity to clean up and shut down as well.

GoldMineshutDown

The GoldMineShutDown event indicates when the GoldMine application is shutting down. It has no parameters. Following is an example of implementing the GoldMineShutDown event in VB.NET using a delegate function. For an example implementing an event handler in Visual Basic 6.0, see the VetoWindow event for the .

```
Private Sub GMShutdown()
MsgBox("GoldMine has closed", MsgBoxStyle.Information, "XML API")
End Sub

Private Function CreateGMEventHandler() As Boolean
Try
'Here we try to setup an eventhandler for goldmine shutdown
'if we set this up before we're logged in it launches the api
'and mucks things up, here we create the varriable, and
'assign it an event
```

```
Dim GMEvent As New GoldMine.GMSystemEvents
AddHandler GMEvent.GoldMineShutDown, AddressOf GMShutdown
Catch ex As Exception
Return False
End Try
Return True
End Function
```

Business Logic Methods

Overview

GoldMine introduces *Business Logic*, a concept to simplify and streamline product integration with GoldMine. Business Logic transactions wrap commonly used procedures into a single call. For example, to attach a new detail to a record, you simply execute the WriteDetail function.

Business Logic Functions and Name/Value Pairs

To make these Business Logic methods useful, developers need a mechanism for passing multiple parameters to the various methods. GoldMine provides a set of functions to control Name/Value containers in the `GMXS32.DLL`, described in . Alternatively, all of the business logic functions are accessible via the GoldMine XML API. The XML API uses all of the same business logic function names and data names (Name/Value pairs).

This chapter describes the Business Logic methods available. These methods may be called from the GMW_ Execute function (`GMXS32.DLL`) or via the GoldMine XML API (`GMXMLAPI.DLL`).

Controlling Database Session Handling

The SetSessionHandling function controls the way GoldMine handles database sessions. The default, the safest method, is to open and close sessions for each request. This can be changed to increase performance to keep sessions open. The function accepts one name/value pair, KeepOpen. Its possible values are 1 or 0. The function returns one name/value pair, OldState, with possible values of 1 or 0, so you know what was previously set prior to your change. Finally, the function returns a status of either 0 on failure, or 1 on success. This function applies only to the `GMXS32.DLL`.

Creating or Updating a Contact Record

WriteContact creates or updates a contact record. If RecID is passed as null, then a record will be created. Otherwise, the record will be updated. You may also create a new contact record with a RecID you provide. This function will respect record curtaining and will not update areas of the contact record that the logged-in user does not have permission to change. Contacts created through this function will have the Automated Process marked to be attached to new records.

GoldMine API Version: 5.00.041

Required Name/Value Pairs

RecID is the record ID of the record to update. If null, a record will be created, unless the ExternRecID or ExternAccNo name/value pairs are included.

Optional Name/Value Pairs

Any valid Contact1 or Contact2 field.

Special Name/Value Pairs

WriteContact Special NV Pairs

Name	Description
Email	E-mail address profile value. Additional e-mail addresses may be added to the contact record by including this name/value pair with an existing RecID. Cannot update any e-mail addresses with this function. See UpdateEmailAddress. Only one address will be marked as primary. If additional addresses are added through this function, they will not be primary unless the next name/value pair is set.
PrimaryEmail	Indicates to mark the specified e-mail address as primary. Set to 1 to mark primary.
WebSite	Web site detail value. Additional Web sites may be added to the contact record by including this name/value pair with an existing RecID. Cannot update any Web sites with this function. See UpdateWebSite.
NonUSAPhone	International phone format is used if NonUSAPhone = 1, Default is 0.
WebUserName	Web username to assign to this contact. For details, see “ContactLogin.”
WebPassword	Web password to assign to this contact. For details, see “ContactLogin.”
ExternRecID	User-supplied RecID to be used for a new record. RecID name/value pair must be empty to use this functionality.
ExternAccNo	User-supplied AccountNo to be used for a new record. RecID name/value pair must be empty to use this functionality.

Output Name/Value Pairs

WriteContact Output NV

Record	Description
RecID	If new record created.
AccountNo	AccountNo of the record

WriteCONTACT Error Codes

WriteContact Error Codes

Code	Description
	Success
	General Failure
-1	Incomplete request to create based on external RecID
-2	Could not create a new record
-3	Could not create a new record based on external RecID.
-4	Could not commit to disk
-5	No access or could not lock record
-6	Record does not exist.
-7	External RecID already exists on this system.

Updating an E-mail Address

UpdateEmailAddress is used to update the value of an existing e-mail address detail record.

GoldMine API Version: 5.50.10111

Required Name/Value Pairs

UpdateEmailAddress Required NV Pairs

Name	Description
RecID	RecID of the e-mail record to be modified
NewAddress	New address to write

Optional Name/Value Pairs

UpdateEmailAddress Optional NV Pairs

Name	Description
Accountno	Accountno of the contact the e-mail address is associated with.
MIME	Set to “1” to use MIME when sending to this address.
RTF	Set to “1” to use RTF when sending to this address.
Primary	Set to “1” to mark this updated e-mail address as primary.
Wrap	Set to “1” to wrap lines when sending to this address.

Updating a Web Site Record

The UpdateWebSite function is used to update the value of a Web Site detail record.

GoldMine API Version: 5.50.10111

Name/Value Pairs

UpdateWebSite NV Pairs

Name	Description
RecID	Web site record RecID—required
NewSite	New Web site value to write—required
Primary	Set to “1” to mark this Web site as the primary Web site for the contact record

Updating Notes of a Primary Contact Record

WriteContactNotes updates the Notes of a primary contact record and appends the proper header information to the top of the Note. If both AccountNo and RecID are passed, only AccountNo will be used. The Note header will use the current date/time and default to the logged-in user name.

GoldMine API Version: 5.00.041

Required Name/Value Pairs

WriteContactNotes Required NV Pairs

Name	Description
Notes	Note text to add
AccountNo	AccountNo of the Contact1 record to which to add notes. Not required if RecID is used.
AccountNo	AccountNo of the Contact1 record to which to add notes. Not required if RecID is used.
RecID	RecID of the contact1 record to which to add notes. Not required if AccountNo is used.

Optional Name/Value Pairs

UserID is the UserID used in the note header.

Output Name/Value Pairs

None.

Creating or Updating a Note in a Table

WriteNote creates or updates a note in the table provided by the parameter. The Note header uses the current date/time and defaults to the logged-in user name.

GoldMine API Version: 5.00.041

Required Name/Value Pairs

WriteNote Required NV Pairs

Name	Description
Note	Note text to write.
AccountNo	AccountNo of the contact associated with the note. Required for a new contact. For updates, if it is passed and different than the existing AccountNo, it is checked for validity.
NotesRecID	The Notes table RecID. Required for updates; returned on new notes.

Optional Name/Value Pairs

WriteNote Optional NV Pairs

Name	Description
Table	Table the note is associated with. Values can be OPMGR, CASES, CASE_RESOLUTION, or CONTACT1. Defaults to CONTACT1. Can also pass Notes table abbreviated versions OP, CA, CR, or C1.
LOPRECID	RecID for the associated table's row. Required for all new contacts except for contact notes, because CONTACT1 > RecID is read when checking if the user has access.
UserID	User to attach the note to.
OPPROJTYPE	Record type in case the table parameter is OPMGR or OP. Can be O for Opportunity or P for Project.

Output Name/Value Pairs

None.

WriteNote Error Codes

WriteNote Error Codes

Code	Description
1	Success
0	No PNV passed.
-1	No note passed to write.
-2	Could not find the Notes table. RecID passed.
-3	Contact is new, yet no AccountNo was passed, or bad LOPRECID.

-4	User doesn't have access rights to the contact, or AccountNo is invalid.
-5	Could not initialize a new record in the Notes table.
-6	Unable to open the Notes table.

Creating or Updating an Additional Contact Record

WriteOtherContact creates or updates an additional contact record. If RecID is null, then a record will be created; otherwise, the record will be updated. When RecID is passed as null, an AccountNo should be passed; otherwise, an unlinked record will be created. In addition, a new additional contact may be created using a unique, user-supplied RecID. If the logged-in user does not have master rights and the contact record associated with the additional contact record is curtained, then no new additional contact records or modifications will be allowed.

GoldMine API Version: 5.00.041

Required Name/Value Pairs

None.

Optional Name/Value Pairs

WriteOther ContactNotes Optional NV Pairs

Name	Description
RecID	RecID of the record to update. If null, a record will be created.
ExternRecID	User-supplied RecID to be used for a new additional contact. The RecID and ExternRecID name/value pairs are mutually exclusive. If the RecID pair is supplied, this pair will be ignored.
AccountNo	AccountNo of linked Contact1 record
Contact	Contact name
Title	Title
Ref	Reference
Dear	Salutation
Phone	Phone number
Fax	Fax number
Ext	Extension
Address1	Address Line 1
Address2	Address Line 2

Address3	Address Line 3
City	City
State	State
Zip	ZIP Code
Country	Country
Notes	Notes
LinkAcct	Link Account RecID

Special Name/Value Pairs

WriteOtherContact Special Name/Value Pairs

Name	Description
Email	E-mail address of the additional contact
NonUSAPhone	Set to 1 for a nonUSA phone format
UseMergeCodes	Set to 1 if you want to set the **Use Merge Codes** option
MergeCodes	Merge codes

Error Codes

WriteContact Error Codes

Code	Description
1	Success
0	General Failure
-1	It will be a duplicate
-2	Couldn't create external record
-3	Couldn't find or lock the record
-4	Couldn't write to the database
-5	No access to the contact linked to this record

Output Name/Value Pairs

RecID returns the new RecNo or RecID if a new record was created.

Creating or Updating a Detail Record

WriteDetail creates or updates a detail record. If RecID is null, then a record will be created; otherwise, the record will be updated. When a RecID is passed as null to create a record, an AccountNo should be passed; otherwise, an unlinked record will be created. In addition, a new detail record may be created using a unique, user-supplied RecID. If the logged-in user does not have master rights and the contact record associated with the detail record is curtained, then no new detail records or modifications will be allowed.

GoldMine API Version: 5.00.041

Required Name/Value Pairs

Detail is the name of the detail.

Optional Name/Value Pairs

WriteDetail Optional NV Pairs

Name	Description
RecID	RecID of the record to update. If null, a record will be created.
ExternRecID	A user-supplied RecID to be used for a new detail record. The RecID and ExternRecID name/value pairs are mutually exclusive. If the RecID pair is supplied, this pair will be ignored.
AccountNo	AccountNo of linked Contact1 record.
Ref	Value of the detail being created or updated.
Notes	Notes for the detail record.

Special Name/Value Pairs

UField 1–Ufield 8 correspond to the extended detail fields; that is:

UField1	UField5
UField2	UField6
UField3	UField7
UField4	UField8

Output Name/Value Pairs

RecID returns the new RecNo if a record was created.

Error Codes

WriteDetailError Codes

Name	Description
	Success
	General Failure
-1	It will be a duplicate
-2	Couldn't create external record
-3	Couldn't find or lock the record
-4	Couldn't write to the database
-5	No access to the contact linked to this record

Creating or Updating a Linked Document

WriteLinkedDoc creates or updates a linked document record. If RecID is null, then a record will be created; otherwise, the record will be updated. When RecID is passed as null, an AccountNo should be passed; otherwise, an unlinked record will be created.

GoldMine API Version: 5.00.041

Required Name/Value Pairs

RecID is the RecID of the record to update. If null, a record will be created.

Optional Name/Value Pairs

Optional NV Pairs

Name	Description
AccountNo	AccountNo of linked Contact1 record.
FileName	Full path and filename.
Ref	Title of the document.
Notes	Notes

Special Name/Value Pairs

SyncFile synchronizes the file with remote sites if set to 1.

Output Name/Value Pairs

RecID returns the new RecNo if a record was created.

Error Codes

These error codes were added in GoldMine API Version: 5.70.20222

WriteLinkedDoc Error Codes

Name	Description
	Success
	General Failure
-1	Contact not found
-2	Access denied
-3	Could not add the linked document
-4	Requested linked document does not exist
-5	Could not write the linked document
-6	The given accountno does not match the existing one

Creating or Updating a Referral

WriteReferral creates or updates a referral from one contact record to another.

GoldMine API Version: 5.00.041

Required Name/Value Pairs

RecID is the RecID of the record to update. If null, a record will be created.

Optional Name/Value Pairs

WriteReferral Optional NV Pairs

Name	Description
FromAccNo	AccountNo of the 'From' referral.
ToAccNo	AccountNo of the 'To' referral.
FromRef	Reference line for the 'From' record.
ToRef	Reference line for the 'To' record.
Notes	Notes
AppendNotes	Appends Notes with a time stamp. You must pass a valid RecID.

Special Name/Value Pairs

OppSummary is a 12-bit flag of opportunity summary check boxes in the Referrals properties. This is a sequence of twelve 1s or 0s.

Output Name/Value Pairs

RecID returns the new RecNo if a Record was created.

Creating or Updating Activities

WriteSchedule creates or updates a scheduled activity record. If RecID is null, then a record will be created; otherwise, the record will be updated. When RecID is passed as null, an AccountNo should be passed; otherwise, an unlinked record will be created.

GoldMine API Version: 5.00.041

Required Name/Value Pairs

RecID is the RecID of the record to update. If null, a record will be created.

Name	Description
AccountNo	AccountNo of linked Contact1 record
RecType	RecType. For a list of valid RecTypes, see the table structures for CAL.
CaseRecID	The Case record ID to link to the calendar event. You cannot attach a case and an opportunity/project to the same event.
LOPRECID	The opportunity or project to attach the event to. It cannot be used with a case recid.
UserID	User name of activity
Contact	Contact name
Ref	Reference: line
Notes	Notes
ActvCode	Activity code
OnDate	Date of activity (Required for scheduling recurring activities when using `gm6s32.dll` – GoldMine 6.0)
OnTime	Time of activity (Required for scheduling recurring activities when using `gm6s32.dll` – GoldMine 6.0)
Duration	Duration of activity
Alarm	If set to 1, an alarm will set for the specified user. Default is 0.
AlarmDate	Date of alarm. Must set Alarm to 1 to use.
AlarmTime	Time of alarm. Must set Alarm to 1 to use.
RSVP	If set to 1, the activity will be sent with an RSVP. Default is 0.
Private	If set to 1, the activity will be marked as private. Default is 0.
Notify	If set to 1, the scheduled user will receive a notification. Default is 0.

Amount	Sale amount. Only used when RecType = S
ProbSale	Probability of sale. Only used when RecType = S
UnitsSale	Number of units in sale. Only used when RecType = S
ccUsers	List of additional users to schedule the activity for
bccUsers	List of users to inform about the activity througha GoldMine e-mail.
Resources	List of resources to reserve for this activity.
RecurType	*Use only for versions of GoldMine earlier than 6.0.* For recurring activities. Specify one of the following to indicate how the activity should be repeated: ***Value Description*** 1070 Daily 1071 Weekly 1072 Bi-weekly 1073 Monthly 1074 Quarterly 1075 Yearly 1076 Every n days. Also use RecurNDays nv pair. 1080 First. Also use RecurOnDays nv pair. Ex. Schedule on the first Monday of every month. 1081 Second. Also use RecurOnDays nv pair. 1082 Third. Also use RecurOnDays nv pair. 1083 Fourth. Also use RecurOnDays nv pair. 1084 Last. Also use RecurOnDays nv pair.
RecurNDays	*Use only for versions of GoldMine earlier than 6.0.* Recur every x days. Used when RecurType is set to 1076.

RecurOnDay	*Use only for versions of GoldMine earlier than 6.0.* Used when RecurType is set to 1080-1084. For example, you wish the activity to be schedule for the first Monday of every month, then RecurType would be set to 1080 and RecurOnDay would be set to 1092. ***Value Description*** 1091 Sunday 1092 Monday 1093 Tuesday 1094 Wednesday 1095 Thursday 1096 Friday 1097 Saturday
RecurSkipWeekend	*Use only for versions of GoldMine earlier than 6.0.* Set to 1 (default) if the activities should not be scheduled on weekends, should the scheduling pattern call for it to land on one. Otherwise 0.
RecurFromDate	*Use only for versions of GoldMine earlier than 6.0.* The date to begin scheduling the activities.
RecurToDate	*Use only for versions of GoldMine earlier than 6.0.* The date to end the scheduled activities.

GoldMine 6.0 NV Pairs

The following WriteSchedule NV pairs are specific to GoldMine versions 6.0 and greater. They apply to scheduling recurring activities. The NV pairs for the previous versions of GoldMine are still valid, though in order to implement extended recurrence patterns, these new pairs need to be used in lieu of the previous pairs. If your application will only be used on GoldMine 6.0 systems, it is recommended to use the newer recurrence NV pairs listed below.

Optional WriteSchedule NV Pairs

Name	Description
RecurType	For recurring activities. Specify one of the following to indicate how the activity should be repeated: ***Value Description*** 1 Hourly 2 Daily 3 Weekly 4 Monthly 5 Yearly

RecurFormat	Set to 1 (default) to specify an UNTIL recurrence rule (defined by a start date/time and end date/time) and is used in conjunction with RecurToDate. Set to 2 to specify a COUNT recurrence rule (defined by a start date/time and an integer representing the number of occurrences) and is used with RecurCount.
RecurCount	Represents the number of occurrences at which to bound the range (Used when RecurFormat = 2, omit if RecurFormat = 1).
RecurToDate & RecurToTime	Use to specify the end of the date and time range for scheduling recurring activities. (Used when RecurFormat = 1, omit if RecurFormat = 2)
RecurInterval	Represents how often the recurrence rule repeats
RecurOnDay	The day(s) when the recurrence occurs: The following seven values can be used when RecurType equals 3 through 5. The values can be combined using the bitwise AND operator. ***Value Description*** 1 Sunday 2 Monday 4 Tuesday 8 Wednesday 16 Thursday 32 Friday 64 Saturday The following values should only be used when RecurType is equal to monthly (4) or yearly (5). ***Value Description*** 200 Weekday 201 Weekend Day 202 Day
RecurMonthDay	The day of the month the activity should occur. Values 1 through 31 are valid. Should only be used if RecurType is monthly (4) or yearly (5). If RecurMonthDay is used, then RecurPos is ignored.
RecurPos	Specifies if the activity should be scheduled on the first, second, third, fourth or fifth day specified in RecurOnDay (as in, first Monday of each month, etc). Used only when RecurType is monthly (4) or yearly (5). If RecurMonthDay is set also, this value will be ignored.
RecurMonth	Specifies which month the recurring activity is to be scheduled in when the RecurType is set to monthly (5). Valid values are 1 through 12 and correspond to months respectively (1 = January).

RecurSkipWeekend	Skip weekends when scheduling recurring activities. Valid values or 1 (default) or 0. Use when RecurType is daily (2), monthly (4), or yearly (5).
RecurSkipNon WorkdayHours	Skip hours that are not designated as part of the workday (ex: 5pm through 8 am). Valid values are 1 (default) or 0. Use when RecurType is set to hourly (1).

Output Name/Value Pairs

RecID returns the new RecID if a record was created.

Error Codes

These WriteSchedule error codes were added in GoldMine API Version: 6.0.21021

WriteSchedule Error Codes

Name	Description
	Success
	General Failure
-10	Ondate > RecurEndDate
-11	No Ondate specified
-12	No RecurToTime (or RecurCount)
-13	No weekdays selected in the weekly pattern
-14	Not enough NV Pairs specified

Creating or Updating a History Record

WriteHistory creates or updates a history record, or completes a scheduled activity record. If RecID is null, then a record will be created; otherwise, the record will be updated. When RecID is passed as null, an AccountNo should be passed; otherwise, an unlinked record will be created. To complete a scheduled activity, you must pass CalRecID.

GoldMine API Version: 5.00.041

Required Name/Value Pairs

RecID is the RecID of the record to update. If null, a record will be created.

WriteHistory Optional Name/Value Pairs

WriteHistory Optional NV Pairs

Name	Description
AccountNo	AccountNo of linked Contact1 record.

RecType	RecType. For a list of valid RecTypes, see the table structures for CONTHIST.
UserID	User name of activity
Contact	Contact name
Ref	Reference line
Notes	Notes
ActvCode	Activity code
ResultCode	Result code
OnDate	Date of activity
OnTime	Time of activity
Duration	Duration of activity

WRITE HISTORY Special Name/Value Pairs

WriteHistory Special NV Pairs

Name	Description
CalRecID	RecID of the scheduled activity (Cal table).
Success	If set to 1, the activity was successful. Default is 1.
Private	If set to 1, the activity is marked as private. Default is 0.
RSVP	If set to 1, an RSVP is scheduled. Default is 0.
Link	If Set to 1 indicates that it is linked to the contact record specified in AccountNo.
Amount	Sales amount. Used where RecType = S
ProbSale	Probability of sale. Used where RecType = S
UnitsSale	Number of units in sale. Used where RecType = S

Output Name/Value Pairs

RecID returns the new RecNo if a record was created.

Creating or Updating a Case Record (GoldMine 8.0 or higher)

WriteCase creates or updates a Case for the GoldMine 8.0 service module.

Required Name/Value Pairs

The following fields are required for new records.

Name	Description
Accountno	Accountno of the contact to link with the Case
Number	The case number - required for new - alpha numeric 40 chars

Optional Name/Value Pairs

WriteCase Optional NV Pairs

Name	Description
Recid	A valid Case record ID to modify, passed only on a modify call. Required for updates.
Accountno	Accountno of the contact to link with the Case
Number	The case number - required for new - alpha numeric 40 chars
User	The goldmine user name to assign the case to. If not passed, assumed to be the logged in user.
IsTemplate	Use this case as a template - 1 = template, 0 = not
IsRead	Has been read 1= true, 0 = false
Status	A numeric representation of the status. 0 = <unknown>, 1 = assigned, 2 = reassigned, 3 = escalated, 4 = resolved, 5= abandoned, 6 = open, 7 = closed
Priority	A priority code created by the users. Alpha numeric 40 chars
Source	The source of the case - alpha numeric 40 chars
Category	A category code created by the user - Alpha numeric 40 chars
Type	Type code created by the user - alpha numeric 40 chars
Offering	A data field mainly used to list what you've offered to the case subject 200 chars
Subject	A short description reference 200 chars
Description	A long description of the case issues and steps
Notes	Notes on a given resolution - see htmlnotes and appendnodes
ResolutionType	A user defined resolution code - alpha numeric 40 chars
ResolutionNotes	Notes on the resolution - See htmlnotes and appendnotes
DueDate	The date that resolution is due. The format must be date then time in your locale's format (3-16-07 10:00 am)
ResolvedBy	The goldmine user that resolves the issue

ResolvedDate	The date of actual resolution. The format must be date then time in your locale's format (3-16-07 10:00 am)
HTMLNotes	Boolean that determines if the notes passed are pre formatted for HTML. 1= true, 0 = false, default is 0
AppendNotes	Boolean that determines if notes are overwritten or a new note is appended to the end. 1= append, 0 = overwrite. Default is 1

Error Codes

WriteCase Error Codes

Code	Description
1	Success
0	No NV container passed
-1	Required NV pairs not passed
-2	Valid case id not passed
-3	Could not open Cases table
-4	Could not find CaseID
-5	Could not open CaseTeamLink table
-6	Could not initialize new record
-7	Attempt to append new record failed

Output Name/Value Pairs

RecID returns the RecID in a name-value container if a new record was created.

Creating or Updating a Case Attachment (GoldMine 8.0 or higher)

WriteCaseAttachment creates or updates a CaseAttachment.

Required Name/Value Pairs

The following fields are required for new records: CaseID, RecType, Describes, Title and Location. See the following table for details.

Optional Name/Value Pairs

WriteCaseAttachment Optional NV Pairs

Name	Description
RecID	A valid CaseAttachment table recid to modify, passed only on a modify call. Required for updates.
CaseID	A valid Case table recid to attach the file or link to. Required if new.
RecType	The recType, an integer of 0 or 1. 0 = File, 1 = Link. Required if new.
Describes	An integer of 0 or 1. 0 = Problem, 1 = Solution. Required if new.
Title	The title for the file - Alpha numeric 100 chars. Required if new.
Location	The URI for the file or link. Alpha-numeric 512 chars. Required if new.

Error Codes

WriteCaseAttachment Error Codes

Code	Description
1	Success
0	No NV container passed
-1	New with invalid case id
-2	New and missing required values
-3	Could not open Cases table
-4	Could not find CaseID in case table
-5	Couldn't open CaseAttachement table
-6	Could not init new record or find and lock the record to be modified
-7	Invalid rectype passed
-8	Invalid describes value passed

Output Name/Value Pairs

RecID returns the RecID in a name-value container if a new record was created.

Adding a GoldMine User as a Case Team Member (GoldMine 8.0 or higher)

WriteCaseTeamLink adds a GoldMine user as a Team member for a case.

Required Name/Value Pairs

WriteCaseTeamLink NV Pairs

Name	Description
CaseID	A valid Case table recid to add the user. Required for updates.
UserName	The GoldMine User Name to add to the Case Team
Role	The role for the user. User defined alpha numeric 40 chars.

Error Codes

WriteCaseTeamLink Error Codes

Code	Description
1	Success
0	No NV container passed
-1	New with invalid case id
-2	New and missing required values
-3	Could not open Cases table
-4	Could not find CaseID in case table
-5	Could not open CaseAttachement table
-6	Could not init new record or find and lock the record to be modified
-7	Invalid rectype passed
-8	Invalid describes value passed

Output Name/Value Pairs

RecID returns the RecID in a name-value container if a new record was created.

Attaching an Automated Process

AttachTrack attaches an automated process to a contact record.

GoldMine API Version: 5.00.041

ATTACHTRACK Required Name/Value Pairs

Required NV Pairs

Name	Description
AccountNo	AccountNo of the contact record (Contact1) to which to attach the track.
Track	

UserID

Output Name/Value Pairs

RecID returns the new RecNo if a record was created.

Executing an SQL Query

SQLStream executes a SQL query and returns the data in a DataStream. For details, see .

GoldMine API Version: 5.00.041

Required Name/Value Pairs

SQL is the SQL statement to execute.

Optional Name/Value Pairs

SQLStream Optional NV Pairs

Name	Description
Filter	Xbase filter expression.
FldDlm	Field delimiter. Defaults to CR.
RecDlm	Record delimiter. Defaults to LF.
StartRec	Starting record. Defaults to 1.
GetRecs	Maximum records to return. Defaults to 100.
MaxBufSize	Maximum buffer size. Defaults to 32k.
Raw (XML API ONLY)	Indicates the format the data should be returned as. The default ("0") puts the data into XML format. Setting Raw to "1" returns the data stream in the old return packet format, as described below.

Output Name/Value Pairs

Output is the return DataStream.

The packet header (the first 12 characters of the Output NV pair) record consists of two sections:

First byte can be 0, 3, or 4:

0 indicates that more records are available, which could be fetched with another SQLStream call (be sure to set the StartRec nv pair to one more than the number of records returned in the first call)

3 indicates the end-of-file (EOF)

4 indicates the beginning-of-file (BOF)

Number following the dash indicates the total number of data records contained in the packet.

If the Raw parameter is set to 0 using the GoldMine XML API, the packet will be XML formatted. See the XML Return Packet for information on interpreting this data format.

NOTE: If the return DataStream is too large for the specified buffer size, SQLStream returns a value of -5. When the buffer in increased to an adequate size, SQLStream will return the data in a DataStream. The practical upper limit for buffer size is 2 MB. If your query returns data in excess of 2 MB, we recommend using DS_Query and DS_Fetch rather than SQLStream for better performance

Creating a Cont act Group

The CreateContactGroup function is used to create an empty contact group. Members are then added through the AddContactGrpMembers function. For details, see "Adding Contacts to a Contact Group" on page .

GoldMine API Version: 5.70.20222

Required Name/Value Pairs

GroupName is the name of the group to be created.

Optional Name/Value Pairs

CreateContactGroupOptional NV Pairs

Name	Description
GroupCode	Group code.
UserName	Group owner. The currently logged in user will be used if empty.
SyncGroup	1 (default) if the group should be synced. Otherwise 0.

Output Name/Value Pairs

CreateContactGroup Output NV Pairs

Name	Description
GroupNo	Group number of the created group. Use this to add members through the AddContactGrpMembers function.

Return Codes

CreateContactGroup Return Codes

Code	Description
1	Success
0	General Failure
-1	Missing group name
-2	Could not create the group

Adding Contacts to a Contact Group

Once a contact group is created with CreateContactGroup, the AddContactGrpMembers function is used to add contacts to that group. In addition, this function can be used to add members to existing groups.

GoldMine API Version: 5.70.20222

Required Name/Value Pairs

AddContactGrpMembers Optional NV Pairs

Name	Description
GroupNo	Group number.
Members	Multi value NV pair containing multiple NV pair containers. Each container stores information for each contact to add to the group. See below for details of the child containers.

Members NV Pair Child Container Name/Value Pairs

Members NV Pairs

Name	Description
Accountno	Accountno of the member to add
Reference	Reference of the member.
Sort	Sort value for the member

Members NV Pair Child Container Output Name/Value Pairs

Members Output NV Pairs

Name	Description
MemberNo	Recno/recid of the member record

Output Name/Value Pairs (parent container)

AddContactGrpMembers Output NV Pairs

Name	Description
MembersAdded	Number of members added.

Return Codes

Note that on the first instance the function encounters an error adding a member, it will stop adding members and not continue through the list of requested members.

AddContactGrpMembers Return Codes

Code	Description
	Success
	General Failure
-1	Missing Group Number
-2	Unable to find group
-3	Cannot add member
-4	No members added

Using AddContactGrpMembers

Below are the steps you should take in order to populate the Members Name/Value pair correctly.

1. Create parent container using GMW_NV_Create.
2. Populate GroupNo Name/Value pair in parent container.
3. Create another container using GMW_NV_Create to serve as the child container (assign to a different long variable).
4. Populate any common Name/Value pairs in the child container (i.e. Reference).
5. Loop through the contacts you want to add and do the following:
   - Assign Accountno name/value pair in the child container.
   - Assign any other optional name/value pairs in the child container (i.e. reference or sort).
6. Use the GMW_NV_AppendNvValue function to copy the contents of the child container to a new container within the Members name/value pair of the parent container:

   ```
   GMW_NV_AppendNvValue (lParentGMNV, "Members", lChildGMNV)
   ```
7. Execute WriteSchedule.

Reading a Record

ReadRecord reads a record from the specified table, based on RecID. When the TableName=Contact1, all Contact2 fields will also be returned. Any record that is inaccessible through GoldMine due to record curtaining will not be returned. Any fields inaccessible through GoldMine due to field-level access restrictions will not be returned.

GoldMine API Version: 5.00.041

Required Name/Value Pairs

ReadRecord Required NV Pairs

Name	Description
TableName	GoldMine table to read.
RecID	RecID of the Contact1 record to return.

Optional Name/Value Pairs

Address Block returns the address as one block of text instead of in separate fields for Address1, Address2, City, State, and so on, when equal to 1.

Special NVs

AccountNo can be used to find the record instead of RecID if TableName=Contact1.

Output Name/Value Pairs

All field values for the specified record.

ReadRecord Output NV Pairs

Name	Description
Email	Returns the primary e-mail address if TableName=Contact1.
Website	Website profile will return if TableName=Contact1.
CurtainingState	Indicates level of curtaining for returned record. 0 – none, 1 – partial, 2- full. Use this to save a call to IsContactCurtained.

Return Codes

ReadRecord Return Codes

Code	Description
1	Success
0	General Failure
-1	No access to the record
-2	Record not found
-3	Invalid parameters

Reading a Contact1 or Contact2 Record

ReadContact reads a contact record from Contact1 and Contact2. Any record that is inaccessible through GoldMine due to record curtaining will not be returned. Any fields inaccessible through GoldMine due to field level access restrictions will not be returned.

GoldMine API Version: 5.00.041

Required Name/Value Pairs

RecID is the RecID of the Contact1 record to return.

Optional Name/Value Pairs

AddressBlock returns the address as one block of text instead of in separate fields for Address1, Address2, City, State, and so on, when equal to 1.

Special NVs

AccountNo can be used to find the record instead of RecID if TableName=Contact1.

Output Name/Value Pairs

All Contact1 and Contact2 field values.

ReadContact Output NV Pairs

Name	Description
Email	Returns the primary e-mail address if TableName=Contact1.
Website	Website profile will return if TableName=Contact1.
CurtainingState	Indicates level of curtaining for returned record. 0 = none, 1 = partial, 2 = full. Use this to save a call to IsContactCurtained.

Return Codes

ReadContact Return Codes

Code	Description
1	Success
0	General Failure
-1	No access to the record
-2	Record not found
-3	Invalid parameters

Returning Alerts Attached to a Contact Record

GetContactAlerts returns all alerts attached to a contact record.

GoldMine API Version: 5.00.041

Required Name/Value Pairs

GetContactAlerts Required NV Pairs

Name	Description
RecID	RecID of the Contact1 record to return. You can optionally use AccountNo.

AccountNo	AccountNo of the Contact1 record. You may optionally use RecID.

Output Name/Value Pairs

The function returns the number of contact alerts in the AlertsCount Name/Value. For each alert, the function returns five fields. Each set of alert fields has the alert number appended to the field name (represented by X in the following table).

GetContact Alerts Output NV Pairs

Name	Description
AlertsCount	Number of alerts.
CodeX	Three-character alert code.
DescX	80-character description.
NotesX	64k of RTF alert message (optional).
CreatorX	User that assigned the alert.
SaveHist	Value of 1 indicates that GoldMine will save a history record when the user acknowledges the alert.

Return Codes

GetContactAlerts Return Codes

Code	Description
0	No PNV or no alerts found.
>0	The number of alerts returned.

Attaching an Alert

SetContactAlert attaches an alert to the specified contact record. To generate an alert list, execute the GetAllAlerts function.

GoldMine API Version: 5.00.041

Required Name/Value Pairs

SetContactAlert Required NV Pairs

Name	Description
RecID	RecID of the Contact1 record to which to attach this alert. You can optionally use AccountNo.
AccountNo	AccountNo of the Contact1 record. You can optionally use RecID.

Code	Three-character Alert Code.
Creator	Creator of the Alert.
SaveHist	A history record is generated each time the Alert is acknowledged if set to 1.

Output Name/Value Pairs

None.

The GMW_Execute function will return the following values:

GMW_ExecuteReturn Values for SetContactAlert

Return	Description
0	Contact not found
1	Alert is added
2	Alert is already attached

Returning All Alerts

GetAllAlerts returns all alerts defined within GoldMine.

GoldMine API Version: 5.00.041

Required Name/Value Pairs

None.

Output Name/Value Pairs

The function returns the number of contact alerts in the AlertsCount name value. For each alert, the function returns five fields. Each set of alert fields has the alert number appended to the field name (represented by X below):

GetAllAlerts Data Fields Returned

Name	Description
AlertsCount	Number of alerts.
CodeX	Three-character alert code.
DescX	80-character description.
NotesX	64k of RTF alert message (optional).

Returning a User List

GetUsersList returns a list of all GoldMine users.

GoldMine API Version: 5.00.041

Required Name/Value Pairs

None.

Output Name/Value Pairs

GetUsersList Required NV Pairs

Name	Description
UserList	Comma-delimited list of all user names
UserCount	Number of users in the list
UserGroupsList	Comma-delimited list of user groups
UserGroupsCount	Number of user groups

The GMW_Execute function will return the same value as UserCount.

Returning a User Group Member List

GetGroupUsersList returns a list of all members of a GoldMine user group.

GoldMine API Version: 5.00.041

Required Name/Value Pairs

GroupNo is the user group number. See the GetUsersList or GetUserMemberships functions for information on how to retrieve a UserGroupsList and their numbers.

Output Name/Value Pairs

GetGroupUsers List Output NV Pairs

Name	Description
UserList	Comma-delimited list of all user names
UserCount	Number of users in the list

The GMW_Execute function will return the same value as UserCount.

Returning Group Memberships for a Specified User

GetUserMemberships returns a list of all user group memberships for the specified UserID.

GoldMine API Version: 5.00.041

Required Name/Value Pair

UserID is the GoldMine user name.

Output Name/Value Pairs

GetUserMemberships Output NV Pairs

Name	Description
UserGroupsList	Comma-delimited list of user group numbers of which the user is a member
UseGroupsCount	Number of users in the list

The GMW_Execute function will return the same value as UserGroupsCount.

Saving a User Group

WriteGroupUsersList saves the user members to a user group. You must have Master Rights to execute this function.

GoldMine API Version: 5.00.041

Required Name/Value Pairs

WriteGroup UsersList Required NV Pairs

Name	Description
GroupNo	User group number. For details on retrieving a UserGroupList name and number, see the GetUsersList or GetUserMemberships functions.
UserList	Comma-separated list of users who are members of the specified group.

Output Name/Value Pair

UserCount is the number of updated user records.

The GMW_Execute function will return the same value as UserCount.

Retrieving the Names of User Groups

GetGroupName returns the descriptive names given for a comma-delimited list of group numbers.

GoldMine API Version: 5.50.10111

Required Name/Value Pairs

GetGroupNameRequired NV Pairs

Name	Description
GroupList	Comma-delimited list of group number for which to retrieve names (for example: 1,4,5,8)

Return Name/Value Pairs

GetGroupNameReturn NV Pairs

Name	Description
GroupCount	Number of groups actually found
Each Group Number	The corresponding name for the group number specified as the value

Example

GroupCount = 4

1 = MyGroup

2 = Techs

3 = Sales

4 = Management

Evaluating an Xbase Expression on a Contact Record

XbaseContactExpr parses a contact- related Xbase expression and return the result and type of the expression. It is possible to parse multiple expressions in one call.

GoldMine API Version: 5.50.10111

Name/Value Pairs

XbaseContactExprNV Pairs

Name	Description
AccountNo	Account number of the contact to parse against
XbaseExpr	Expression to parse, or
ExprCount	Number of expressions to parse, and
XBaseExpr1 .. XBaseExprN	Expressions to parse

Returned Name/Value Pairs

XbaseContactExpr Returned NV Pairs

Name	Description
Result	Result of parsing the expression

Type	Type of the expression. Possible values: 0 – Error 1 – Number 2 – String 3 – Date 5 – Bool, or
Result1 . . ResultN	Result of each expression
Type1 .. TypeN	Type of each expression—see type above for possible values

Return Values

The XbaseContactExpr function returns the following status values:

XbaseContractExpr return values

Value	Description
-2	Contact was not found
-1	No accountno given
0	No expression
1..N	Number of correctly parsed expressions

Encrypting Text

The EncryptString function encrypts a plain text string to a Base64 ASCII encoded buffer.

GoldMine API Version: 5.50.10111

Required Name/Value Pairs

EncryptString Required NV Pairs

Name	Description
Key	Key to use. This can be any value.
ClearText	Text to encrypt.
HashKey	Set to “1” to specify the key to be hashed before use. Provides better security if the key is very simple.

Returned Name/Value Pairs

EncryptStringReturned NV Pairs

Name	Description
CryptText	Encrypted string in an ASCII encoded buffer (Base 64).

Decrypting Encoded Text

The DecryptString function decrypts encoded text.

GoldMine API Version: 5.50.10111

Required Name/Value Pairs

DecryptStringRequired NV Pairs

Name	Description
Key	Key to use. Must be the same as when encrypting.
CryptText	Text to decrypt.
HashKey	Set to "1" to specify the key to be hashed before use. Provides better security if the key is very simple.

Returned Name/Value Pairs

DecryptString Returned NV Pairs

Name	Description
ClearText	Decrypted string. The text is padded with spaces to be on a 64-bit (8 bytes) boundary.

Retrieving the Default Contact Automated Process

Within GoldMine, a user can specify a particular Automated Process (AP) to be attached to new contact records. The GetNewContactAP function returns the RecID of the Automated Process that is assigned to automatically attach to new records. The NV Pair in which the Automated Process RecID is returned is called NewContactAP. The function returns 1 on success, and 0 on failure.

Deleting Calendar Items

The DeleteSchedule function is used to delete scheduled activities.

GoldMine API Version: 5.50.10111

Required Name/Value Pair

DeleteSchedule Required NV Pair

Name	Description
RecID	RecID of the scheduled item to delete (Cal record RecID)

Return Values

Value	Description
0	OK
-1	Empty or bad RecID value
-2	Can't open database
-3	Cal record not found
-4	Failed to delete
-9999	General exception (unknown error)

Deleting History Items

The DeleteHistory function is used to delete completed activities.

GoldMine API Version: 5.50.10111

Required Name/Value Pairs

DeleteHistory Required NV Pairs

Name	Description
RecID	RecID of the history item to delete (ContHist record ID)

Return Values

Value	Description
0	OK
-1	Empty or bad RecID value
-2	Can't open database
-3	ContHist record not found
-4	Failed to delete
-9999	General exception (unknown error)

Handling GoldMine Security

An important part of your integration considerations should be how you will handle the security of your GoldMine database. All business logic functions that write and read from the GoldMine database adhere to the security settings for the user logged in through GMW_LoadAPI or GMW_LoadBDE. Additional functions are provided to aid in managing GoldMine security.

Creating a New GoldMine Login

WriteGMUser enables you to create GoldMine user names. The user logged into the API must have master rights.

GoldMine API Version: 5.50.10111

Name/Value Pairs

WriteGMUser NV Pairs

Name	Description
UserName	Username to add
Password	Password for the user
FullName	Full name of the user
SQLUser	SQL login to be used for this user if connecting to an MS SQL database
SQLPassword	Password for the SQL login
MasterUser	Set to “1” to enable master rights for this user, otherwise “0”

Return Values

WriteGMUser returns “1” on success and “0” on failure.

Reading a GoldMine Login

The ReadGMUser function returns detailed information about a GoldMine Login.

GoldMine API Version: 6.00.21021

Output Name/Value Pairs

ReadGMUserNV Pairs

Name	Description
UserName	Username to add.

Password	Password for the user
FullName	Full name of the user
SQLUser	SQL login to be used for this user if connecting to an MS SQL database
SQLPassword	Password for the SQL login
MasterUser	

Return Values

ReadGMUser returns “1” on success and “0” on failure.

Retrieving Security Access

GetUserAccess returns the security information specified for the currently logged-in user.

GoldMine API Version: 5.50.10111

GetUserAccess Return Name/Value Pairs

Name	Description
SQLUser	SQL Username specified for this user
Master	Whether or not the user has master rights: 1 master, 0 not

AccessRights	This name/value pair consists of a set of flags indicating the access rights the user has to various areas of GoldMine. Each permission is either granted or denied based on the value of its position in the set of flags. A value of "1" signifies the permission is granted, and "0" if it is denied. Below is a chart of the positions in the set of flags and their corresponding permission: *Position Permission* 2 Others Calendar 3 Others History 4 Others Forecasts 5 Others Reports 6 Others Forms 7 Others Filters 8 Others Groups 9 Others Linked Documents 12 Create new contact records 13 Edit Fields 14 Delete contact records 15 Assign contact record owners 16 Edit tab folders 17 Schedule automated processes 19 Issue SQL Queries 20 Netupdate 21 Output To menu 25 Build groups 35 Real time tab 36 Toolbar settings
UsersCALENDAR	The user group's calendar that this user has permission to view. Valid if permission is set. See AccessRights name/value pair.
UsersHISTORY	The user group's history that this user has permission to view. Valid if permission is set. See AccessRights name/value pair.
UsersLINKS	The user group's linked documents that this user has permission to view. Valid if permission is set. See AccessRights name/value pair.
UsersGROUPS	The user group's contact groups that this user has permission to view. Valid if permission is set. See AccessRights name/value pair.
UsersREPORTS	The user group's reports that this user has permission to view. Valid if permission is set. See AccessRights name/value pair.

UsersFILTERS	The user group's filters that this user has permission to view. Valid if permission is set. See AccessRights name/value pair.
UsersFORMS	The user group's forms that this user has permission to view. Valid if permission is set. See AccessRights name/value pair.
UsersSALES	The user group's sales that this user has permission to view. Valid if permission is set. See AccessRights name/value pair.
ForceLogoutAt	The time (AM/PM) that this user will be forced to exit GM.
IdleLogout	The amount of time (in minutes) that GM will remain idle before shutting down.
MenuExclusion	A string containing the menu ID's that are excluded from the user's instance of GM, delimited by an underscore. Ex. "344_531_164_"
NewRecOwnership	A Boolean value that states whether or not new users are automatically assigned to this user.

Retrieving Field-Level Access Rights

FieldAccessRights returns a list of all fields and the access right for the logged-in user for each.

GoldMine API Version: 5.50.10111

Required Name/Value Pairs

FieldAccessRightsOutput NV Pairs

Name	Description
TotalFieldCount	Number of fields returned
Field Names (for example, COMPANY, CONTACT, KEY1)	Possible values: N - No Access R - Read Access W - Read/Write Access

Example NV Container Returned from FieldAccessRights

```
**TotalFieldCount** = 3
COMPANY = R
CONTACT = W
ACCOUNTNO = N
```

Retrieving Visible Fields

NonCurtainedFields returns a \n delimited list of fields visible on partially curtained records. The list is returned in the NonCurtainedList and SemiPartNonCurtainedList name/value pairs. The latter pair indicates which fields are visible when the contact record is semi-partially curtained (all four top quadrants of the contact record are visible) and is only returned in GoldMine 6.0 and greater.

NOTE: You must pass an empty NV container with all calls that do not take any parameters.

Checking for Record Curtaining

IsContactCurtained tests a contact record for curtaining.

Required Name/Value Pairs

IsContactCurtained Required NVPairs

Name	Description
RecID	Record ID of the Contact1 record to test. AccountNo can be passed in place of this Name/Value pair.
AccountNo	AccountNo of the Contact1 record to test. RecID can be passed in place of this Name/Value pair.

Output Name/Value Pair

Curtain NV pair return values

Value	Description
0	Not curtained
1	Partial curtaining
2	Fully curtained

The GMW_Execute function will return TRUE if the record was found.

Generating a Remote License File

CreateRemoteLicense generates a license file for a remote user or site. The resulting license.dbf (6.7 or lower) or license.bin (7.0 or higher) file will be stored in a subdirectory off a specified path. If the path specified is C:\temp, then the file will be in C:\temp\user where “user” is the GoldMine username provided to the function.

GoldMine API Version: 5.50.10111

Name/Value Pairs

CreateRemoteLicense Required NV Pairs

Name	Description
UserName	User or site name
LicPath	Location to place the license files. If left empty, the file will be put in a directory called UserLic under the sysdir (GoldMine directory)
LicType	U (undocked) or S (site)

SiteUsers	For a sublicense site, the number of users at that site

Return Name/Value Pairs

CreateRemoteLicense returns one NV pair called “Result” with the following return codes. This code is also returned as the function’s result value.

CreateRemoteLicense Return Result Codes

Value	Description
1	OK
0	General Error
-1	No Username
-2	User already undocked
-3	Cannot open user file
-4	User not found
-5	Undocked license count exceeded
-8	Cannot create the new license file

Removing a Remote License

RemoveRemoteLicense removes an undocked user or sub-license site.

GoldMine API Version: 5.50.10111

Name/Value Pairs

RemoveRemoteLicense NV Pairs

Name	Description
UserName	User Name or Site Name
LicType	U (undocked) or S (sublicense site)

Return Name/Value Pairs

RemoveRemoteLicense returns one NV pair called “Result” is returned with the following return codes. This code is also returned as the function’s result value.

RemoveRemoteLicense Return Result Codes

Value	Description
1	Success

0	General Error

E-mail Name/Value Functions

This set of functions allows the manipulation of GoldMine and Internet e-mail.

Reading a Mail Message

The ReadMail function reads an e-mail message based on either the RecID in the Mailbox table or the Cal/ContHist tables. A flag is required to specify whether the function should look in the Cal tables or ContHist tables. The mail message can be opened for editing or reading.

GoldMine API Version: 5.50.10111

Required Name/Value Pairs

None.

Optional Name/Value Pairs

ReadMail Optional NV Pairs

Name	Description
MboxRecID	Mailbox RecID. Either this NV pair or the RecID NV pair must be included.
RecID	Cal/History RecID.
History	Flag identifying location of RecID provided. 1 for History, 0 or nothing for Cal.
ForEdit	1 to open for editing, 0 or nothing if for reading.
Password	Password to decrypt the message if it was encrypted on send.

READMAIL Output Name/Value Pairs

Output NV Pairs

Name	Description
RecID	Cal/History RecID
MboxRecID	Mailbox RecID

MailboxFlags	Collection of flags: MAILBOX_ITEM_READ 0x0001 MAILBOX_ITEM_HIST 0x0002 MAILBOX_ITEM_OUTBOUND 0x0004 MAILBOX_ITEM_ATTACH 0x0008 MAILBOX_ITEM_REDIRECT 0x0010 MAILBOX_ITEM_GMASLINKS 0x0020
To	List of all the To: recipients. Comma-delimited and quoted if needed.
Cc	List of all the CC: recipients. Comma-delimited and quoted if needed.
Bcc	List of all the Bcc recipients. Comma-delimited and quoted if needed.
ReplyTo	Reply to address (if any)
From	The from address of the message. Will usually be the default user account, but can contain other addresses.
Subject	Subject
Org	Organization that will appear in the header.
MessageID	MessageID from the header.
Status	Message status from the header.
Date	Internet standard date from the header.
XMailer	XMailer from the header.
OtherHeaders	Other headers not categorized above.
Body	Message body. This will be different in edit mode.
Attachments	A question mark delimited list of attachments.
Alarm	1 if set, 0 if not.
History	1 if from History, 0 if not.
Private	1 if private, 0 if not.
RSVP	1 if marked for RSVP, 0 if not.
ReturnReceipt	1 if requested, 0 if not.
Encrypted	1 if the message is encrypted, 0 if not.
Outgoing	Message is an outgoing message (queued for delivery or already sent): 1 or 0.

MailType	Following types are possible: SMM_Internet 0 This is the one to handle SMM_GoldMine 1 Only exists for compatibility with GoldMine 4.0 SMM_Template 2 Template mails.
IsMIME	1 if MIME based message, 0 if not.
AccountNo	Accountno of the linked contact (or empty).
LinkedContact	If an additional contact is linked this will have the ContSupp RecID.
LinkedOppty	RecID of the linked opportunity or project (if applicable).
Activity	Activity Code
Result	Result Code
CalDate	Calendar/History date
CalTime	Calendar/History time
Contact	Contact name
CreateBy	User who created the mail or "Internet" if the message was retrieved from the mail server.
Folder	Folder in which the message is stored.
SubFolder	Subfolder in which the message is stored. No value will be returned if the message(s) already exist in the Inbox or Outbox.
RecType	RecType of the Calendar record: In Cal: Q = Queued mail, M = Incoming In History: MI = Incoming, MO = Outgoing
Reference	Calendar/History reference. Usually initialized from the subject automatically.
User	User who owns the message belongs.
HasTransferSet	1 if the e-mail message has a transfer set attached to it, 0 if not.
HasVCard	1 if the e-mail message has a Vcard attached to it, 0 if not.
HasWebImport	1 if the e-mail message has a WebImport attached, 0 if not.

Return Codes

ReadMailReturn Result Codes

Value	Description
1	Success

0	Failure
-1	Message is private
-2	Message not found, or cannot be loaded
-3	Exception

Queuing a Message for Delivery

The QueueMail function queues a message for delivery. The actual delivery is not handled through the DLL. It is recommended to set up a specific user in GoldMine responsible for sending multiple users' mail on a regular basis.

If the message to be queued already exists within GoldMine, pass either the Mailbox RecID or the Calendar/History RecID with the history flag. When queuing a new message, do not provide values for the RecID name/value pairs or the flag.

GoldMine API Version: 5.50.10111

QueueMail Optional NV Pairs

Name	Description
MboxRecID	The mailbox RecID. Either this NV pair or the RecID NV pair must be included.
RecID	The Cal/History RecID.
History	Flag identifying location of RecID provided. 1 for History, 0 or nothing for Cal.
To	A list of To: addresses delimited by commas and double-quoted as needed
Cc	A list of CC addresses delimited by commas and double-quoted as needed
Bcc	List of Bcc addresses delimited by commas and double-quoted as needed
ReplyTo	Reply-to address
OtherHeaders	Special headers, if needed
Organization	Organization field
From	From address
Subject	Subject of the message.
BodyText	Body text
TextRTF	Set to non-zero if the text should be in RTF format
NumAttachments	Number of attachments to send
Attachment0..AttachmentN	Indexed list of attachments. The first attachment NV pair will be Attachment0, then Attachment2, and so on.

MailboxFlags	See ReadMail
AccountNo	Accountno of the contact to which the message is linked
OpptyRecID	RecID of an opportunity or project to which the message should be linked
LinkedContact	RecID of the contsupp record of an additional contact, if so linked
ActivityCode	Activity code
CalDate	Calendar date – the date to actually send the message
CalTime	Calendar time – the time to actually send the message
Reference	Reference in the calendar record
Result	Result code
User	User name
Private	1 to mark as Private, 0 if not
RSVP	1 to request a RSVP, 0 if not
Alarm	1 set alarm, 0 if not
ReturnReceipt	Request a return receipt. The value portion of the pair should be the return address to which to send the receipt.
SaveAsDraft	Set to 1 if the message should be saved as a draft and not queued.
UseMIME	Set to 1 to force the message to be a MIME message even if no attachments are available, otherwise 0.
AttachVCard	Set to 1 to attach the user Vcard to the message, otherwise 0
SendNow	Set to “1” to send the message immediately without queuing it. Pertains to a GoldMine user only (no Internet recipients).
Password	Specify a password to set this message to be encrypted. See also the EncryptUSMode name/value pair.
EncryptUSMode	Set to “1” and specify a password to use the US encryption mode. This will be forced to “0” if the license does not allow it.

Return Name/Value Pairs

QueueMail Return NV Pairs

Name	Description
RecID	Calendar/History RecID

MboxRecID	Mailbox RecID
MailBoxFlags	Mailbox flags (see above for description)

Updating a Mail Message

The UpdateMail function allows the modifying of the opportunity with which the mail is associated and indicates whether the message has been read, its encryption state, and whether or not it is private.

GoldMine API Version: 5.50.10111

Required Name/Value Pairs

UpdateMail Required NV Pairs

Name	Description
MboxRecID	Mailbox RecID. Either this NV pair or the RecID NV Pair must be included
RecID	Cal/History RecID
History	Flag identifying the location of RecID provided. 1 for History, 0 or nothing for Cal.

Optional Name/Value Pairs

UpdateMail Optional NV Pairs

Name	Description
OpptyRecID	Opportunity with which the message is associated.
Private	Set to 1 to mark the message as private, otherwise 0.
MarkRead	Set to 1 to mark the message as having been read, 0 for unread.
Password	Password to decrypt the message.
EncryptUSMode	Set to 1 for 128-bit encryption, 0 for 32-bit encryption.

Saving a Mail Message into GoldMine

The SaveMail function enables you to save a mail message into GoldMine when the actual sending or retrieval of the message took place in an outside application. The folder/subfolder specified to save the message to will be created by GoldMine if needed. There's no need to create it beforehand.

GoldMine API Version: 5.50.10111

The NV Pairs coincide with the QueueMail function. SaveMail also has the following additional NV pairs:

Optional Name/Value Pairs

SaveMail Optional NV Pairs

Name	Description

OutgoingMail	Set to 1 if mail was sent by the user. Don't include, or set to 0, if it was received mail
Folder	The name of the folder in which to put the mail. If nothing is given, it will be put in the Inbox or Outbox according to the OutgoingMail NV pair
SubFolder	The name of the subfolder in which to put the mail. Folder must also be defined. To put it in a sub-inbox, set Folder to "X-GM-INBOX"

Return Codes

The SaveMail function returns the following values:

SaveMail Return Codes

Value	Description
	Cannot initialize
-1	Cannot queue the message
-2	Can't save the message (for incoming e-mail)
-3	Can't complete the message to the requested folder
-4	An existing message was loaded. SaveMail works only with new messages.

Deleting a Message

The DeleteMail function deletes a message according to the settings specified for the user within GoldMine (use trashcan or not, delete attachments or not). A message can be deleted based on either the Mailbox RecID or the Calendar/History RecID with a flag to tell the function if it should look in the Calendar or History table.

GoldMine API Version: 5.50.10111

Required Name/Value Pairs

DeleteMailRequired NV Pairs

Name	Description
MboxRecID	Mailbox RecID for the record to be deleted, or
RecID	Calendar/History RecID
History	1 if the RecID in the RecID NV pair is from the History table, or 0 if from the Calendar table

Filing a Message in History

The FileMail function files a mail message in history specified by the Mailbox table RecID.

GoldMine API Version: 5.50.10111

Required Name/Value Pairs

FileMail Required NV Pairs

Name	Description
MboxRecID	Mailbox RecID for the record to be deleted

Optional Name/Value Pairs

FileMail Optional NV Pairs

Name	**Description**
Folder	Folder to file into
Subfolder	Subfolder to file into
Result	Result to be marked in history
ToUser	Used to specify another username if filed on behalf of that user

Return Codes

FileMail Return Codes

Value	Description
	Success
	General Failure
-1	Cannot initialize Internet-related structs
-2	Message doesn't exist or can't be loaded
-3	Cannot complete the message or the message is already filed

Preparing the NV Container for a New Mail Message

A number of options and templates are available to GoldMine users for sending e-mail within the GoldMine program. For new messages being sent through the API, all of these can be accessed by utilizing the PrepareNewMail function. This function will return a container containing the same NV pairs returned by the ReadMail function reflecting the appropriate settings within GoldMine. You may then modify the container accordingly and send the message with QueueMail.

GoldMine API Version: 5.50.10111

Required Name/Value Pairs

None.

Optional Name/Value Pairs

PrepareNewMailOptional NV Pairs

Name	Description
LinkToAccount	AccountNo of the contact to link the new message to.
LinkToAddContact	RecID of the additional contact record to link to. LinkToAccount must also be specified.
ManualTo	Specific e-mail address to send to.
MailType	Pass a 1 to indicate creation of an internal GoldMine mail message.

Return Name/Value Pairs

Same as ReadMail

Preparing the NV Container to Reply to a Mail Message

A number of options and templates are available to GoldMine users for sending e-mail within the GoldMine program. All of these can be accessed for replying to messages sent through the API by utilizing the PrepareReplyMail function. In addition, the body text of the message may be returned containing quoted text from the message being replied to. This function will return a container containing the same NV pairs returned by the ReadMail function reflecting the appropriate settings within GoldMine. You may then modify the container accordingly and send the message with QueueMail.

GoldMine API Version: 5.50.10111

Required Name/Value Pairs

PrepareReplyMail Required NV Pairs

Name	Description
FromRecID	RecID from Cal or ContHist of the message replied to
FromHist	1 if the message is in History (contHist), otherwise assumed to be in Cal
QuoteText	Text to quote in the reply. If this NV pair is left empty, the full message text will be quoted. If so, set in the user's mail preferences.
ReplyToAll	Reply to all recipients of the original message, not just the sender
ToEMail	Set to 0 if replying to a non-mail activity

Optional Name/Value Pairs

PrepareReplyMailOptional NV Pairs

Name	Description

LinkToAccount	AccountNo of the contact to whom to link the new message.
LinkToAddContact	RecID of the additional contact record to link to LinkToAccount must also be specified.

Return Name/Value Pairs

Same as ReadMail—see .

Preparing an NV Container to Forward a Mail Message

A number of options and templates are available to GoldMine users for sending e-mail within the GoldMine program. For forwarded messages being sent through the API, all of these can be accessed by using the PrepareFwdMail function. In addition, PrepareFwdMail includes the original message body text and header information to be forwarded. This function will return a container containing the same NV pairs returned by the ReadMail function reflecting the appropriate settings within GoldMine. You may then modify the container accordingly and send the message with QueueMail.

GoldMine API Version: 5.50.10111

Required Name/Value Pairs

PrepareFwdMail Required NV Pairs

Name	Description
FromRecID	RecID from Cal or Conthist of the message replied to
FromHist	1 if the message is in History (conthist), otherwise assumed to be in Cal
Redirect	Pass a 1 to create a redirected mail instead of forwarded.
ForwardToGMUser	Set to 1 to forward the mail to a GoldMine user instead of another contact record.
FwdToUser	If ForwardToGMUser is set, then set to the desired GoldMine username to forward the message to.

Optional Name/Value Pairs

PrepareFwdMail Optional NV Pairs

Name	Description
LinkToAccount	Accountno of the contact to link the new message to.
LinkToAddContact	RecID of the additional contact record to link to. LinkToAccount must also be specified.

Return Name/Value Pairs

Same as ReadMail—see .

Adding an E-mail Center Folder

Use AddFolder to create a folder and/or subfolder in the **E-mail Center**. If both the folder and the subfolder do not exist, then both will be created.

GoldMine API Version: 5.50.10111

Name/Value Pairs

AddFolder NV Pairs

Name	Description
Folder	Folder name to be created—Required
SubFolder	Optional subfolder name
User	Optional user name. Defaults to the logged-in user

Deleting an E-Mail Center Folder

Use DeleteFolder to remove folders or subfolders from the *E-Mail Center*. If both a folder and subfolder are supplied, only the subfolder will be deleted. Any messages included in the specified folder are also deleted.

GoldMine API Version: 5.50.10111

Name/Value Pairs

DeleteFolder NV Pairs

Name	Description
Folder	Folder name—Required
Subfolder	Optional subfolder name.

Obtaining a List of E-Mail Center Folders

The FolderList function returns a sorted list of folders from the **E-Mail Center**. Folders are returned with a prefix of “0” if the folder is a top-level folder, or a prefix of “1” if it is a subfolder. System folders are not returned, only user folders.

GoldMine API Version: 5.50.10111

Return Name/Value Pairs

FolderList Return NV Pairs

Name	Description
FolderCount	Number of folders in the list

Folder1..FolderN	List of folders

Example List of Folders

```
FolderCount = 6
Folder1 = 0Filed
Folder2 = 1January 2000
Folder3 = 2February 2000
Folder4 = 0Sent
Folder5 = 1January 2000
Folder6 = 2February 2000
```

FromList

The FromList function returns a list of unique From addresses to use in outgoing e-mail.

GoldMine API Version: 5.50.10111

Return Name/Value Pairs

FromList Return NV Pairs

Name	Description
FromCount	Number of From addresses returned
From0..FromN	List of addresses, indexed from 0 to FromCount-1
History	Flag identifying the location of RecID provided. 1 for History, 0 or nothing for Cal

Accessing E-mail Templates

The TemplateList function returns a list of e-mail templates for a specified user.

GoldMine API Version: 5.50.10111

Optional Name/Value Pairs

TemplateList Optional NV Pairs

Name	Description
User	Username for whom to get the list of templates. Default is the currently logged-in user
IncludePublic	Set to “1” to include public templates

Return Name/Value Pairs

TemplateList Return NV Pairs

Name	Description
TemplateCount	Number of templates in the list.

Name1..NameN	Names of the templates, indexed from 0 to TemplateCount-1.
RecID1..RecIDN	RecIDs of the templates, indexed from 0 to TemplateCount-1.

Retrieving E-mail Account Information

The GetAccountsList function returns a set of name/value pairs describing all e-mail accounts defined for the currently logged-in user. Because a user may have multiple e-mail accounts defined, the name/value pairs are indexed to identify the account that corresponds to the setting. The index number is appended to the beginning of each name. The indexes begin with zero (0).

GoldMine API Version: 5.50.10111

Return Name/Value Pairs

GetAccountsList Return NV Pairs

Name	Description
AccountsCount	Number of accounts
DefaultAccountID	Default account number
Indexed Name/Value Pairs:	
AccountID	ID needed by the other e-mail account-related functions (for example, OnlineList)
DisplayName	Name of the e-mail account displayed in the **E-mail Center**. If available, the account name is used, and if the user requests that mailto:user@server will always be shown, then they're appended to the account name.
User	User to whom the profile is assigned (same as the logged-in user)
AccountName	User-defined descriptive name given to the e-mail account
POP3Server	Address of the POP3 server
Username	Username for the POP3 server
Password	Password for the POP3 account
OwnUser	User who owns the account. This is used so one user can retrieve e-mail for another user. The result is that e-mail messages retrieved by JOHN but with OwnUser set to MARY, will appear in MARY's e-mail center, not in JOHN's.
POPAuthMode	POP server's authentication mode. Possible values: 0 – PASS 1 – APOP 2 – RPA 3 – NTLM

DeleteMail	Set to “1” to auto-delete mail from this account, otherwise “0”
AutoRetrieve	Set to “1” to auto-retrieve messages from this account, otherwise “0”
UseSigFile	Set to “1” to use a signature file with this account, otherwise, “0”
SigFile	Path and filename to the signature file if UseSigFile is set
POPPort	POP3 Server’s port number
TOPSupport	Set to 1 if the account supports the TOP command
ShowInIMC	Set to “1” to show this account in the Internet Mail Center
SMTPServer	SMTP Server address
ReturnAddress	Return e-mail address for this account
SMTPPort	Port number for the SMTP server
SMTPUser	Username for the SMTP server, if the server requires authentication.
SMTPPass	Password for the SMTP server, if the server requires authentication
SMTPAUTH	Set to “1” if the SMTP server requires authentication
SMTPAUTHMode	Possible Values: 0 – None 1 – Login 2 - NTLM

Retrieving a List of Messages Waiting Online

The OnlineList function returns a list of all messages waiting online for the requested account. Each message’s corresponding NV pairs are indexed from 1 to N according to the number of available messages. The index numbers are appended to the end of the NV pair name.

GoldMine API Version: 5.50.10111

Required Name/Value Pairs

OnlineList Required NV Pairs

Name	Description
AccountID	AccountID to retrieve. Get this value from GetAccountsList.

Return Name/Value Pairs

OnlineList Returned NV Pairs

Name	Description

Error	Will include an error message if an error occurred and there is a message to present (like server error messages).
NumMessages	The number of messages available online.
Indexed Name/Value Pairs:	
Message_Subject	Subject of the message.
Message_DispDate	Date as displayed in the GoldMine E-mail Center.
Message_Date	Date in the message.
Message_Time	Time the message was sent.
Message_Address	Address that sent the message.
Message_Size	Size in bytes.
Message_DispSize	Size as displayed in GM.
Message_Type	Possible Values: 0 – Plain 1 – Plain MIME (no attachments) 2 – Complex MIME 3 – GM Sync set
Message_AccNo	Accountno to which this message is linked.
Message_UID	Server UID of this message.
Message_Num	Message number on the server—use for retrieval/delete.
Message_Mailer	Mailer that generated the message.
Message_ReplyTo	Reply-to address for this message.
Message_To	Address to which the message is sent.
Message_CC	CC (copy) addresses for the message.
Message_Bcc	Bcc (blind copy) addresses for the message.
Message_GMUsersTo	Comma-delimited list of GoldMine users to whom the message is being sent.
Message_GMUsersCc	List of GoldMine users to whom the message is being copied.
Message_Org	E-mail organization field.
Message_OtherHeaders	Other headers associated with this message.
Message_Read	1 if the message has already been read, otherwise 0.
Message_Headers	Formatted headers as they appear in the preview window.

Message_Body	Message body (according to the number of lines previewed in the E-mail Center).

Return Values

OnlineList Return Values

Value	Description
1	Success
0	General Failure
-1	Invalid Account ID
-2	Protocol Error—see the description in error
-3	Comm error—see the description in error
-4	Timeout or other error—see the description in error
-5	Unknown error

Retrieving Messages

The RetrieveMessages function retrieves specified messages that are online. The returned name/value pairs will have a message number appended to the end of the name.

GoldMine API Version: 5.50.10111

Required Name/Value Pairs

RetrieveMessage Required NV Pairs

Name	Description
AccountID	Account ID to use.
AllMessages	Set to “1” for all messages to be retrieved.
MessageList	Tab (\t) delimited list of message numbers (taken from OnlineList) to retrieve.

Return Name/Value Pairs

RetrieveMessage Return NV Pairs

Name	Description
Message_CalRec	Cal RecID of the message, ***** if an error occurred
Message_MboxRec	Mailbox RecID of the message, ***** if an error occurred.

Return Values

RetrieveMessages Return Values

Value	Description
1	Success
0	General Failure
-1	Invalid Account ID
-2	Protocol Error—see the error description in error
-3	Comm error—see the error description in error
-4	Timeout or other error—see the error description in error
-5	Unknown error

Deleting Online E-mail Messages

The DeleteMessages function allows deletions of messages waiting online.

GoldMine API Version: 5.50.10111

Required Name/Value Pairs

DeleteMessages Required NV Pairs

Name	Description
AccountID	Account ID to use.
AllMessages	Set to “1” for all messages to be deleted.
MessageList	Tab (\t) delimited list of message numbers (taken from OnlineList) to delete.

Return Name/Value Pairs

The returned name/value pair will have each message number appended to the end of the name.

GoldMine API Version: 5.50.10111

DeleteMessages Return NV Pairs

Name	Description
Message_Deleted	“1” if the message was deleted successfully.

Return Values

DeleteMessages Return Values

Value	Description
1	Success
0	General Failure
-1	Invalid Account ID
-2	Protocol Error—see the error description in error
-3	Comm error—see the error description in error
-4	Timeout or other error—see the error description in error
-5	Unknown error

Saving a Manual List of Recipients

The SaveManualRcptList function will receive a list of manually provided recipients and save them to an .ini file. The name/value pair list will be Recipient1.RecipientN with the values being the addresses you wish to add to the list. Any missing entry will be saved as an empty address.

GoldMine API Version: 5.50.10111

Retrieving a Manual List of Recipients

The GetManualRcptList function returns a list of the saved manual recipient list. The return value will be “1” for success and “0” for failure. The container will have a name/value pair NumberOfRecipients with the number of recipients. Finally, it will contain Recipient0..RecipientN with the actual addresses.

GoldMine API Version: 5.50.10111

Managing Internet E-mail Preferences

GetEmailPrefs and SetEmailPrefs allow you to get and set the Internet preferences for the user. The preferences correspond with the **Internet Preferences** dialog box within GoldMine. The functions work the same, except the former receives information from GoldMine and the latter updates the data in GoldMine.

GoldMine API Version: 5.50.10111

IMPORTANT: Before calling SetEmailPrefs, the values of the e-mail preferences in the NV pair container must be preloaded with GetEmailPrefs. Otherwise, all e-mail preferences not included in the container for SetEmailPrefs will be deleted from GoldMine.

Optional input (SetEmailPrefs) and Output (GetEmailPrefs) Name/Value Pairs

GetEmailPrefs and SetEmailPrefs Name/Value Pairs

Name	Description
UserName (GoldMine 6.0 or greater ONLY)	The GoldMine user whose e-mail preferences you wish to retrieve or set
MultiActive	1 – Show all accounts in the mail center 0 – Show only the default account
PreviewLines	Number of lines to preview in the E-Mail Center prior to downloading the message
QuoteAll	1 to quote entire message by default when replying, otherwise 0
NewQuoteStyle	1 to specify a custom quote string identifier, otherwise 0
QuoteString	Quote string identifier to be used if NewQuoteStyle is set. Ex: >>
Organization	User-specified signature .txt file
UseOrg	1 to include the signature specified in Organization
SaveHistDefault	1 – Save filed mail in history by default 0 – Do not
AttachDir	Folder in which to save attachments.
OnlyGMMail	1 – When auto retrieving, retrieve only mail from other GoldMine clients. 0 – Auto retrieve mail from all clients
SkipLarge	If automatic retrieval is set, set to 1 to skip large e-mail message larger than size specified in MaxEmailSize, otherwise 0
MaxEmailSize	Limit on size of messages to be automatically retrieved if SkipLarge is set to 1
SkipNoAddress	1 indicates to not skip addresses not on file, otherwise 0
WarnAboutRTF	1 – warn user before sending HTML mail 0 – Do not
GetUnreadMail	If automatic retrieval is set, set to 1 to retrieve only unread mail, otherwise 0
UseHeaderDate	1 to use the date in the mail header, otherwise 0
CompleteOnReply	1 to complete the original message being replied to, otherwise 0
UUEncodeScan	1 to scan mail for UUEncoded Data, otherwise 0
VcardAction	100 if incoming Vcards are not to be saved
Use8BitEncoding	1 to use 8 bit encoding, otherwise 0
AutoSpell	1 to automatically spell check messages before sending, otherwise 0
ForceWrapAt	When forcing line wrap, wrap at this specified column number

WrapReplyAt	Wrap quoted lines in reply at this specified column number
LoadPublicTemplates	1 to show public e-mail templates, otherwise 0
ReadOnGet	1 to Open 'Read E-mail' dialog on retrieval, otherwise 0
LinkOnGet	1 – Prompt user if incoming e-mail address is not on file 0 – Do not
SkipOnDispose	1 – Go to next message in reader after disposing of (deleting/filing) the current one 0 – Close the reader
ShowHeaders	Settings for the mail center preview window headers display: 0 – no headers 1 – summary of headers only 2 – full headers display
UseTrashCan	1 to use trash can for deleted mail, otherwise 0.
EmptyTrashOnExit	1 to empty trash when closing E-Mail Center, otherwise 0.
ConfirmEmptyTrash	1 to confirm before deleting from trash can, otherwise 0.
ShowFullAccountName	1 to show both the e-mail address and the account name (if available) for online accounts, otherwise 0.
DiscardWebImportMessages	1 to discard Web import message after the data has been imported, otherwise 0.
AutoWebImport	1 to import data when retrieving E-Mail Center mail, otherwise 0 (setting this to 0 does NOT assume BackgroundWebImp).
BackgroundWebImp	1 to import data on background e-mail retrieval, otherwise 0 (setting this to 0 does NOT assume AutoWebImport).
SyncContact	Sticky setting from the **E-mail Center** to move the current contact record to the one the selected message belongs to. Set to 1 to activate, 0 otherwise.
KeepOldTransfers	1 to keep the transfer set attachments after retrieving them, otherwise 0.
AllowDeleteAll	1 to enable 'Delete All Server Mail', otherwise 0.
SendVCard	1 to use user-supplied V-card, otherwise 0.
DefaultLinkAddr	When linking an incoming e-mail in GoldMine, if the e-mail does not exist within GoldMine, a dialog box appears to the user. There is a checkbox indicating whether to keep the setting of how the unlinked message is handled. To keep the setting, set this NV pair to 1, otherwise 0.
SyncAttachmentDefault	1 to mark attachments for syncing by default, otherwise 0.
ShowOutlookInIMC	1 to show the **Outlook** folder in the **E-Mail Center**, otherwise 0.

LinkAttachToCont	1 to save attachments as linked documents, otherwise 0.
MarkIncomingAsPrivate	1 to mark incoming messages as private, otherwise 0.
DelAttachWithMsg	1 to delete attachments when deleting the mail, otherwise 0.
KeepUserVCard	Every time GoldMine is restarted and a message is sent, GoldMine creates a VCard for the sending user so that a correct VCard for the user can be sent with the mail if so requested. The VCard is created from information GoldMine has for the logged-in user. Sometimes a user may want to manually edit the VCard to add or change information not available to GoldMine. In this case, the user can ask GoldMine to not recreate the VCard from scratch and GoldMine will use the existing VCard that the user modified. Set to 1 to have GoldMine not create a new VCard, otherwise 0.
BccToSelf	1 to always send a Bcc to the user, otherwise 0.
UseShortDate	1 to use the short date format, 0 to use the long format.
GMAttachAsLinks	1 to send attachments as links to GoldMine users, otherwise 0.
POPIdleDisconnect	Number of minutes to wait without activity only in the **E-mail Center** before automatically disconnecting. The default is 10 minutes.
SkipOverWriteUI	1 to suppress file overwrite prompt, otherwise 0.
RetrieveOverwrite	Default action to be taken when an e-mail attachment file already exists. Possible values: 4 – auto name assignment 5 – do not save the file 6 – overwrite existing file 7 – new file name
DefaultOUTFolder	Folder name under which to put sent mail (replace the default sent folder).
DefaultINFolder	Folder name under which to put filed mail instead of the default **Filed** folder.
MonthlyFolderNames	List of folder names to replace the standard month names used in GoldMine by default. Each month must be * separated and the last entry must be ???*
NewFilingMode (GoldMine 6.0 and greater ONLY)	1 to indicate to use two-level filing mode
ActiveAutoGetMail	1 to activate automatic mail retrieval, otherwise 0.
GetInterval	Frequency in minutes to check for mail automatically, if ActiveAutoGetMail is set.
SendQueueWhen AutoGet	1 to send queued messages when ActiveAutoGetmail is set, otherwise 0.

GetOldToNew	1 to download old messages first, otherwise 0.
UseHTMLByDefault	1 to use HTML when creating new e-mail, otherwise 0.
ExtractEmbedded HTML	1 to extract embedded HTML as attachment, otherwise 0.
TCPTimeout	Number of seconds until a communication timeout.
SendQueueFor	A semicolon-delimited list of GoldMine user names for which this account should send queued e-mail.
FakeSMTPDomain	Used to present the system as a user-defined name if the name returned by the system is not acceptable by the SMTP server.
DefaultTemplate	Specify the default template name for new outgoing messages.
DefaultReplyTemplate	Specify the default template name for new reply messages.
DefaultFwdTemplate	Specify the default template name for new forwarded messages.
Quarantine-to	Name of the quarantine directory to which the quarantine rules move files.

In addition, each e-mail account set up for the user is supplied or returned through a special multi-value item named Profiles. The Profiles NV pair contains a set of containers; each holds information for a different e-mail account. You can determine the number of accounts by calling the GMW_NV_GetMultiValueCount function.

To retrieve the HGMNV pointers for the child containers, call GMW-NV-GetMultiNvValue for each account to retrieve.

If you are setting e-mail preferences, you will want to set the NV values for an e-mail account by using either:

- GMW_NV_AppendNvValue, to copy a prepared container to the Profiles NV pair

or

- GMW_NV_AppendEmptyNvValue, to create an empty child container within the Profiles NV Pair for which you can later set the values.

See “” for more information on these functions.

Profiles child containers have the following NV Pairs.

Profiles Child Container NV Pairs

Name	Description
POP3_Account	The user-editable descriptive name for the account
POP3_Server	The server name or address
POP3_User	The server user name
POP3_Pass	The password for the account

Return_Address	The return address
SMTP_Server	The SMTP server name or address
SigFile	The path to the signature file to use
OwnUser	The user to which this account belongs. This is used so one user can retrieve e-mail for another user. The result is that e-mails retrieved by JOHN but with OwnUser set to MARY will appear in MARY's e-mail center, not in JOHN's.
DelServerMail	Set to 1 to delete the messages from the server upon retrieval, otherwise 0
AutoGetMail	Set to 1 to automatically retrieve mail for this account.
UseSigFile	Set to 1 to use the specified signature file
ShowInIMC	Set to 1 to show this account in the E-mail Center.
UseTOPCmd	Set to 1 if this server supports the TOP command, otherwise 0
POP3_Port	The POP3 server's port number
SMTP_Port	The SMTP server's port number
POP3_AuthMode	The POP server's authentication mode. Possible values: 0 – PASS 1 – APOP 2 – RPA
SMTP_AuthMode	Possible values: 0 – None 1 – Login 2 – NTLM
SMTP_User	The username for the SMTP server, if the server requires authentication
SMTP_Pass	The password for the SMTP server, if the server requires authentication

Validating a Web User Name and Password

ContactLogin validates a WebUserName/WebPassword assigned to a contact.

GoldMine API Version: 5.50.10111

Required Name/Value Pairs

ContactLogin Required NV Pairs

Name	Description
UserName	Contact's Web user name.
Password	Contact's Web password.

Special Name/Value Pairs

ContactLogin Special NV Pairs

Name	Description
NewUserName	Changes the existing Web username. Must be used with NewPassword, and a valid UserName. Password must also be passed for verification.
NewPassword	Changes the existing Web password. Must be used with NewUserName, and a valid UserName/Password must be passed for verification.

Output Name/Value Pairs

ContactLogin Output NV Pairs

Name	Description
AccountNo	Returns the AccountNo of the contact record
RecID	Returns the RecID for the contact record

Notes

This function is useful when writing an extranet solution for GoldMine. To enable GUI access to these features, set ContWebAccess=1 under the [GoldMine] section of your `username.ini`. You can then select *Edit > Record Properties > WebAccess* to set the Web user/pass (maximum of 15 characters each). GoldMine stores Web access data in ContSupp with a RecType of W. Each user name and password must be unique. This information does not synchronize.

Manipulating User-Defined Fields and Views

Beginning in GoldMine 6.00.21021, the ability to read and write changes to the user-defined fields and views was added to the GoldMine API. Most of the following functions use multi-container NV pairs. This means that a single NV pair may contain multiple containers, each with their own set of NV pairs. For example, when reading field views, there will be an NV pair named “View”. This NV pair will contain an entire NV pair container for each field view in GoldMine containing a set of NV pairs that describe that view. In addition, each of those containers will store an NV pair named “Field”. This NV pair will contain an entire NV pair container for each field defined on that view, each with its own set of NV pairs describing that field. For information on how to read and manipulate multi-container NV pairs, please see .

IMPORTANT: The GoldMine user logged into the API must have master rights in order to use these functions.

Reading All Field Views

The GetContactViews function returns all of the field views, including the custom screens, main contact record, and the summary tab fields. As described above, this function utilizes multi-container NV pairs. Execute GetContactViews, passing an empty NV pair container, to retrieve the following NV pairs describing the field views.

GoldMine API Version: 6.00.21021

Output Name/Value Pairs

GetContactViews Output NV Pairs

Name	Description
NumViews	The number of views, including the Main and Summary views.
SelectedViewID	The view currently selected for the Field tab of the contact record.
View	A multi-value list containing a container for each of the actual views. See the table below for details of the NV containers this value stores.

VIEW Name/Value Pairs

The View NV Pair in the container returned by GetContactViews contains NV Pair containers with the following NV Pairs describing the field views defined in GoldMine.

View NV Pair Output Container

Name	Description
ID	The view ID
Name	The view name
TabName	The tab name, if this view has one
UserAccess	The user that is allowed to access this view.
CurrContactset	If set to 1, then the view is visible in the current contact set, otherwise the value is 0
FieldCount	The number of fields this view has.
Field	A multi-value list containing a container for each of the actual fields on the view. See the table below for details of the NV containers this value stores.

Field Name/Value Pairs

The Field NV Pair in the View container contains NV Pair containers with the following NV Pairs describing the fields displayed on the view defined in GoldMine.

Field NV Pair Output Container

Name	Description
VerticalCenter	Y coordinate of the colon on the view
HorizontalCenter	X coordinate of the colon on the view
LabelSize	The length allowed for the label
EditWidth	The width of the editable space for the field on the view
IndexNumber	This is the index associated with this field and is used to decide if the field is searchable (as in the Key fields).
FieldLen	The physical length of the field in the database.
HotKey	Reserved for future use.
TabOrder	The tab order position for the field (the order in which the field will be selected when pressing the tab key)
ExprField	If 1, indicates an expression field, otherwise 0
PhoneFaxField	If 1, indicates if the field is a phone or fax field.
ExtendedProperties	If 1, this field has extended properties
LogInHistory	If 1, any changes made to this field will be logged as a history record on the contact
ReadAccess	Indicates the user or group that can read the contents of the field
WriteAccess	Indicates the user or group that can modify the contents of the field
FieldName	The physical field name
FieldExpr	The field expression if ExprField = 1
GlobalLabel	The global label for the field
LocalLabel	The local label for the field
RecNo	Unique identifier for the field on the view. Needed to modify or delete the field from the view.
LabelExpr	Expression to evaluate to generate the field label
LabelColorExpr	Contains the number representing the color of the label
FieldColorExpr	Contains the number representing the color of the field.
LabelReference	Text value to refer to an expression label (in the list of fields for the view, for example)

GetContactViews Return Values

GetContactViews Return Values

Value	Description
1	Success
0	General Failure
-1	Not a master rights user
-2	Field views cannot be loaded

Deleting a Contact View

The DeleteContactView function deletes the view specified by the view ID. This function accepts one input NV pair, ViewID. Retrieve the ViewID with the GetContactViews function.

GoldMine API Version: 6.00.21021

DeleteContactViews Return Values

DeleteContactViews Return Values

Value	Description
1	Success
0	General Failure
-1	Not a master rights user
-2	Field view cannot be found
-3	The Main and Summary view cannot be deleted
-4	Failed to delete

Creating or Modifying a Contact View

The WriteContactView function enables adding and modifying contact views. In addition, fields displayed on the contact views are added, modified or deleted through this function. This function does not modify the data structure, only the display properties of the fields included in the view.

The input NV container for this function has an NV pair named Field. This is a multi-value NV pair that stores multiple NV pair containers, each describing a field to add, update, or delete on the view. Multiple field operations can be performed in one call to WriteContactView. For example, an existing field could be updated, new fields can be added to the view, and fields can be deleted; each operation has its own Field child container.

GoldMine API Version: 6.00.21021

input Name/Value Pairs

WriteContactView Input NV Pairs

Name	Description
ID	The view ID if updating an existing view. Retrieve this from GetContactViews. Omit if creating a new view.
Name	The view name
TabName	The tab name, if this view has one
UserAccess	The user that is allowed to access this view.
CurrContactset	If set to 1, then the view is visible in the current contact set, otherwise the value is 0
Field	A multi-value list containing a container for each of the field operations to perform (adding, deleting, modifying). See the table below for details of the NV containers to include.

Field Name/Value Pairs

The Field NV Pair in the parent container contains NV Pair containers with the following NV Pairs describing the fields to add, edit or delete from the view.

Field NV Pair Input Container

Name	Description
Action	NEW, UPDATE, or DELETE
RecNo	Unique identifier for the field on the view. Omit if adding a new field to the view. If updating or deleting, retrieve this value by calling GetContactViews.
VerticalCenter	Y coordinate of the colon on the view
HorizontalCenter	X coordinate of the colon on the view
LabelSize	The length allowed for the label
EditWidth	The width of the editable space for the field on the view
HotKey	Reserved for future use.
TabOrder	The tab order position for the field (the order in which the field will be selected when pressing the tab key)
ExprField	If 1, indicates an expression field, otherwise 0
LogInHistory	If 1, any changes made to this field will be logged as a history record on the contact
ReadAccess	Indicates the user or group that can read the contents of the field
WriteAccess	Indicates the user or group that can modify the contents of the field.
FieldName	The physical field name

FieldExpr	The field expression if ExprField = 1
GlobalLabel	The global label for the field
LocalLabel	The local label for the field
LabelExpr	Expression to evaluate to generate the field label
LabelColorExpr	Contains the number representing the color of the label
FieldColorExpr	Contains the number representing the color of the field.
LabelReference	Text value to refer to an expression label (in the list of fields for the view, for example)

WriteContactView output NV pairs

One NV pair is returned, FieldErrors, indicating the number of field-related errors reported. The function continues adding fields even if some fail. For each field the API could not add, an entry is added to the field's child container in an NV pair called Error. The possible values for this pair are:

Field Error Codes

Value	Description
-1	Invalid Action
-2	Requested field not found
-3	No Record ID given for updating or deleting a field
-4	Field cannot be deleted
-5	Field cannot be written
-6	For a new view, only new fields are possible (Action cannot equal MODIFY or DELETE if creating a new view).
-7	Reserved
-8	Reserved
-9	Reserved
-10 -> -20	Invalid positioning

WriteContactView Return Values

WriteContactView Return Values

Value	Description
1	Success

0	General Failure
-1	Not a master rights user
-2	Field view cannot be loaded
-3	Field view could not be saved

Reading Custom Fields

The ReadCustomFields function returns information about the physical properties of custom fields defined in GoldMine. This function contains a multi-value NV Pair, called Field, which stores multiple name/value containers, each with specific details about each field. For information on manipulating and reading multi-value NV pairs, see .

GoldMine API Version: 6.00.21021

ReadCustomFields input NV pairs

ReadCustomFields Input NV Pairs

Name	Description
NumFields	The number of fields returned.
Field	A multi-value NV containing containers for each field returned. See the table below for details on the NV pairs included.

Field NV Pair Container

The Field NV pair in the parent container returned by ReadCustomFields contains an NV pair container for each custom field defined in GoldMine. The fields are described by the following NV pairs:

Field NV Pairs

Name	Description
Description	A text description of the field
Name	The physical field name
Type	The data type stored in the field. Possible values are C (char), D (date), and N (numeric)
Length	The physical length of the field
Decimals	The number of decimal places, if numeric

ReadCustomfields Return Values

ReadCustomFields Return Values

Value	Description

1	Success
0	General Failure
-1	Not a master rights user
-2	Cannot open ContUDef

Modifying the Structure of Custom Fields

The EditCustomField function adds, deletes, or updates a custom field.

IMPORTANT: The API will not rebuild the GoldMine database to reflect the physical changes you may specify with this function. This must be initiated with the GoldMine application.

GoldMine API Version: 6.00.21021

EditCustomField Input NV pairs

EditCustomField Input NV Pairs

Name	Description
Action	NEW, DELETE, or UPDATE
Description	A meaningful description of the field
Name	The field name of an existing field to update or delete. Specify a new unique field name if creating a new field.
Type	The data type of the field: C (char), D (date), or N (numeric)
NewName	The new name to assign to this field if updating an existing one
Length	The physical length to make the field
Decimals	The number of decimals for a numeric field

EditCustomField Return Values

EditCustomField Return Values

Value	Description
1	Success
0	General Failure

-1	Not a master rights user
-2	Cannot open ContUDef
-3	Invalid action
-4	Invalid field name
-5	Name is not unique
-6	Field not found
-7	Field not allowed to be deleted
-8	Invalid field type
-9	Missing field parameters
-10	Failure deleting field
-11	Cannot write record

Reading Calendar Preferences

ReadCalendarPrefs reads a passed user's calendar preferences. If user not passed, assumed to be the session's logged in user. User must be master rights in order to read other's prefs.

READCALENDARPREFS Input NV pairs

ReadCalendarPrefs Input NV Pairs

Name	Description
UserName	The GoldMine user name to read the prefs of

READCALENDARPREFS OUTPUT NV pairs

ReadCalendarPrefs Output NV Pairs

Name	Description
UserName	The GoldMine user name to read the prefs of
UserList	The list of Users that appear on the user's calendar
PegboardUserList	List of users on the user's pegboard
ShowAction	Show actions on the calendar
ShowAppt	Show appointments on the calendar
ShowCall	Show calls on the calendar

ShowEvent	The number of decimals for a numeric field
ShowLitReq	Show literature requests on the cal
ShowMsg	Show msgs on the cal
ShowOccasion	Show occasions on the cal
ShowOpTask	Show opportunity tasks on the cal
ShowOther	Show other events on the cal
ShowProjTask	Show project tasks on the cal
ShowPubEvent	Show public events on the cal
ShowSales	Show sales on the cal
ShowToDo	Show to do's on the cal
ShowHistAction	Show history actions on the cal
ShowHistCall	Show call actions
ShowHistEvent	Show event actions
ShowHistLitReq	Show lit req actions
ShowHistMsg	Show msg actions
ShowHistOpTask	Show op task actions
ShowHistOther	Show other actions
ShowHistProjTask	Show proj task actions
ShowHistPubEvent	Show pub event actions
ShowHistSales	Show sales actions
ShowHistToDo	Show todo actions
DefaultView	The default view of the calendar 0 - day 1 - week 2 - month 3 - year 4 - planner 5 - outline 6 - pegboard
AutoForwardCalls	Automatically forward calls
AutoForwardMsgs	Automatically forward messages

AutoForwardActions	Automatically forward actions
AutoForwardAppts	Automatically forward appointments
AutoForwardSales	Automatically forward sales
AutoForwardOther	Automatically forward other
SyncRecord	Sync the record
ShowTotals	Show totals
ShowIcons	Show icons
RefreshRate	In seconds
PegRefreshRate	Pegboard refresh rate in secs
Color	The windows color value for the cal color
TimeIncrement	In minutes
FontSize	Calendar font size
ShowWeekends	Show weekends
FirstDayofWeek	0 = Sunday 7 = sat
nWeekends	Bit mathed for days to consider the weekend
DayBegin	Military time for the day beginning. 09:00
DayEnd	Day end in military time - 17:00 for 5pm
CalShowActvCode	Show activity code on cal
HistShowActvCode	Show hist activity code
PublishICal	Publish iCal file?
PublishICalPath	The path to where to publish ical - must be in URI format (must start with file:, http:, or ftp:)
PublishICalUser	If path is ftp or http, the login user name
PublishICalPwd	If path is ftp or http, the login user pwd
PublishICalUsersList	The users to publish
PublishIcalAction	Publish actions
PublishIcalAppt	Publish appointments
PublishIcalCall	Publish calls

PublishIcalEvent	Publish events
PublishIcalLitReq	Publish literature requests
PublishIcalMsg	Publish msgs
PublishIcalOccasion	Publish occasions
PublishIcalOpTask	Publish opportunity tasks
PublishIcalOther	Publish other events
PublishIcalProjTask	Publish project tasks
PublishIcalPubEvent	Publish public events
PublishIcalSales	Publish sales
PublishIcalToDo	Publish to do's
PublishIcalHistAction	Publish history actions
PublishIcalHistCall	Publish call
PublishIcalHistEvent	Publish event
PublishIcalHistLitReq	Publish literature request
PublishIcalHistMsg	Publish message
PublishIcalHistOpTask	Publish op task
PublishIcalHistOther	Publish other
PublishIcalHistProjTask	Publish project task
PublishIcalHistPubEvent	publish public event
PublishIcalHistSales	Publish sales
PublishIcalHistToDo	Publish todo
Publish2ICSFilterByDate	Dates to publish
Publish2ICSStartDate	The start date of the range
Publish2ICSEndDate	The end date of the range
PublishICSFilterActivCode	The activity code to filter on
PublishICSFilterRef	The reference code to filter on
PublishICSFilterByLink	Filter on the link? true or false
PublishHTML	Publish cal to HTML?

PublishHTMLPath	The path to where to publish the HTML - must be in URI format (must start with file:, http:, or ftp:)
PublishHTMLUser	If path is ftp or http, the login user name
PublishHTMLPwd	If path is ftp or http, the login user pwd
PublishHTMLUsersList	The users to publish
PublishHTMLAction	Publish actions
PublishHTMLAppt	Publish appointments
PublishHTMLCall	Publish calls
PublishHTMLEvent	Publish events
PublishHTMLLitReq	Publish literature requests
PublishHTMLMsg	Publish msgs
PublishHTMLOccasion	Publish occasions
PublishHTMLOpTask	Publish opportunity tasks
PublishHTMLOther	Publish other events
PublishHTMLProjTask	Publish project tasks
PublishHTMLPubEvent	Publish public events
PublishHTMLSales	Publish sales
PublishHTMLToDo	Publish to do's
PublishHTMLHistAction	Publish history actions
PublishHTMLHistCall	Publish call
PublishHTMLHistEvent	Publish event
PublishHTMLHistLitReq	Publish literature request
PublishHTMLHistMsg	Publish message
PublishHTMLHistOpTask	Publish op task
PublishHTMLHistOther	Publish other
PublishHTMLHistProjTask	Publish project task
PublishHTMLHistPubEvent	Publish public event
PublishHTMLHistSales	Publish sales

PublishHTMLHistToDo	Publish todo
Publish2HTMFilterByDate	Dates to publish 0 - today 1 - yesterday 2 - tomorrow 3 - this week 4 - last week 5 - next week 6 this month 7 last month 8 next month 9 - this year 10 - next year 11 - date range
Publish2HTMStartDate	the start date of the range
Publish2HTMEndDate	the end date of the range
PublishHTMFilterActivCode	the activity code to filter on
PublishHTMFilterRef	the reference code to filter on
PublishHTMFilterByLink	Filter on the link? true or false
PublishFB	publish free busy time if PublishFB is TRUE
PublishFBPath	the path to where to publish free busy - must be in URI format (must start with file:, http:, or ftp:)
PublishFBUser	if path is ftp or http, the login user name
PublishFBPwd	if path is ftp or http, the login user pwd
PublishFBAction	Publish actions
PublishFBAppt	Publish appointments
PublishFBCall	Publish calls
PublishFBEvent	Publish events
PublishFBLitReq	Publish literature requests
PublishFBMsg	Publish msgs
PublishFBOccasion	Publish occasions
PublishFBOpTask	Publish opportunity tasks
PublishFBOther	Publish other events

PublishFBProjTask	Publish project tasks
PublishFBPubEvent	Publish public events
PublishFBSales	Publish sales
PublishFBToDo	Publish to do's
PublishFBHistAction	Publish history actions
PublishFBHistCall	Publish call
PublishFBHistEvent	Publish event
PublishFBHistLitReq	Publish literature request
PublishFBHistMsg	Publish message
PublishFBHistOpTask	Publish op task
PublishFBHistOther	Publish other
PublishFBHistProjTask	Publish project task
PublishFBHistPubEvent	Publish public event
PublishFBHistSales	Publish sales
PublishFBHistToDo	Publish todo
PublishFBFilterByDate	Dates to publish 0 - today 1 - yesterday 2 - tomorrow 3 - this week 4 - last week 5 - next week 6 this month 7 last month 8 next month 9 - this year 10 - next year 11 - date range
PublishFBStartDate	The start date of the range
PublishFBEndDate	The end date of the range
PublishFBFreq	Frequency in minutes

READCALENDARPREFS RETURN VALUES

ReadCalendarPrefs Return Values

Value	Description
1	Success
0	No container passed
-1	Not a master rights user or invalid user name
-2	User ini file doesn't exist
-3	Cannot open the cal table

Modifying Calendar Preferences

WriteCalendarPrefs writes a passed user's calendar preferences. The user must have master rights in order to write another user's preferences.

WRITECALENDARPREFS Input NV pairs

WriteCalendarPrefs Input NV Pairs

Name	Description
UserName	The GoldMine user name to read the prefs of

WRITECALENDARPREFS OUTPUT NV pairs

WriteCalendarPrefs Output NV Pairs

Name	Description
UserName	The GoldMine user name to read the prefs of
UserList	The list of Users that appear on the user's calendar
PegboardUserList	List of users on the user's pegboard
ShowAction	Show actions on the calendar
ShowAppt	Show appointments on the calendar
ShowCall	Show calls on the calendar
ShowEvent	The number of decimals for a numeric field
ShowLitReq	Show literature requests on the cal
ShowMsg	Show msgs on the cal
ShowOccasion	Show occasions on the cal
ShowOpTask	Show opportunity tasks on the cal

ShowOther	Show other events on the cal
ShowProjTask	Show project tasks on the cal
ShowPubEvent	Show public events on the cal
ShowSales	Show sales on the cal
ShowToDo	Show to do's on the cal
ShowHistAction	Show history actions on the cal
ShowHistCall	Show call actions
ShowHistEvent	Show event actions
ShowHistLitReq	Show lit req actions
ShowHistMsg	Show msg actions
ShowHistOpTask	Show op task actions
ShowHistOther	Show other actions
ShowHistProjTask	Show proj task actions
ShowHistPubEvent	Show pub event actions
ShowHistSales	Show sales actions
ShowHistToDo	Show todo actions
DefaultView	The default view of the calendar
AutoForwardCalls	Automatically forward calls
AutoForwardMsgs	Automatically forward messages
AutoForwardActions	Automatically forward actions
AutoForwardAppts	Automatically forward appointments
AutoForwardSales	Automatically forward sales
AutoForwardOther	Automatically forward other
SyncRecord	Sync the record
ShowTotals	Show totals
ShowIcons	Show icons
RefreshRate	In seconds
PegRefreshRate	Pegboard refresh rate in secs

Color	The windows color value for the cal color
TimeIncrement	In minutes
FontSize	Calendar font size
ShowWeekends	Show weekends
FirstDayofWeek	0 = Sunday 7 = sat
nWeekends	Bit mathed for days to consider the weekend
DayBegin	Military time for the day beginning. 09:00
DayEnd	Day end in military time - 17:00 for 5pm
CalShowActvCode	Show activity code on cal
HistShowActvCode	Show hist activity code
PublishICal	Publish iCal file?
PublishICalPath	The path to where to publish ical - must be in URI format (must start with file:, http:, or ftp:)
PublishICalUser	If path is ftp or http, the login user name
PublishICalPwd	If path is ftp or http, the login user pwd
PublishICalUsersList	The users to publish
PublishIcalAction	Publish actions
PublishIcalAppt	Publish appointments
PublishIcalCall	Publish calls
PublishIcalEvent	Publish events
PublishIcalLitReq	Publish literature requests
PublishIcalMsg	Publish msgs
PublishIcalOccasion	Publish occasions
PublishIcalOpTask	Publish opportunity tasks
PublishIcalOther	Publish other events
PublishIcalProjTask	Publish project tasks
PublishIcalPubEvent	Publish public events
PublishIcalSales	Publish sales

PublishIcalToDo	Publish to do's
PublishIcalHistAction	Publish history actions
PublishIcalHistCall	Publish call
PublishIcalHistEvent	Publish event
PublishIcalHistLitReq	Publish literature request
PublishIcalHistMsg	Publish message
PublishIcalHistOpTask	Publish op task
PublishIcalHistOther	Publish other
PublishIcalHistProjTask	Publish project task
PublishIcalHistPubEvent	Publish public event
PublishIcalHistSales	Publish sales
PublishIcalHistToDo	Publish todo
Publish2ICSFilterByDate	Dates to publish
Publish2ICSStartDate	The start date of the range
Publish2ICSEndDate	The end date of the range
PublishICSFilterActivCode	The activity code to filter on
PublishICSFilterRef	The reference code to filter on
PublishICSFilterByLink	Filter on the link? true or false
PublishHTML	Publish cal to HTML?
PublishHTMLPath	The path to where to publish the HTML - must be in URI format (must start with file:, http:, or ftp:)
PublishHTMLUser	If path is ftp or http, the login user name
PublishHTMLPwd	If path is ftp or http, the login user pwd
PublishHTMLUsersList	The users to publish
PublishHTMLAction	Publish actions
PublishHTMLAppt	Publish appointments
PublishHTMLCall	Publish calls
PublishHTMLEvent	Publish events

PublishHTMLLitReq	Publish literature requests
PublishHTMLMsg	Publish msgs
PublishHTMLOccasion	Publish occasions
PublishHTMLOpTask	Publish opportunity tasks
PublishHTMLOther	Publish other events
PublishHTMLProjTask	Publish project tasks
PublishHTMLPubEvent	Publish public events
PublishHTMLSales	Publish sales
PublishHTMLToDo	Publish to do's
PublishHTMLHistAction	Publish history actions
PublishHTMLHistCall	Publish call
PublishHTMLHistEvent	Publish event
PublishHTMLHistLitReq	Publish literature request
PublishHTMLHistMsg	Publish message
PublishHTMLHistOpTask	Publish op task
PublishHTMLHistOther	Publish other
PublishHTMLHistProjTask	Publish project task
PublishHTMLHistPubEvent	Publish public event
PublishHTMLHistSales	Publish sales
PublishHTMLHistToDo	Publish todo
Publish2HTMFilterByDate	Dates to publish 0 - today 1 - yesterday 2 - tomorrow 3 - this week 4 - last week 5 - next week 6 this month 7 last month 8 next month 9 - this year 10 - next year 11 - date range

Publish2HTMStartDate	The start date of the range
Publish2HTMEndDate	The end date of the range
PublishHTMFilterActivCode	The activity code to filter on
PublishHTMFilterRef	The reference code to filter on
PublishHTMFilterByLink	Filter on the link? true or false
PublishFB	Publish free busy time if PublishFB is TRUE
PublishFBPath	The path to where to publish free busy - must be in URI format (must start with file:, http:, or ftp:)
PublishFBUser	If path is ftp or http, the login user name
PublishFBPwd	If path is ftp or http, the login user pwd
PublishFBAction	Publish actions
PublishFBAppt	Publish appointments
PublishFBCall	Publish calls
PublishFBEvent	Publish events
PublishFBLitReq	Publish literature requests
PublishFBMsg	Publish msgs
PublishFBOccasion	Publish occasions
PublishFBOpTask	Publish opportunity tasks
PublishFBOther	Publish other events
PublishFBProjTask	Publish project tasks
PublishFBPubEvent	Publish public events
PublishFBSales	Publish sales
PublishFBToDo	Publish to do's
PublishFBHistAction	Publish history actions
PublishFBHistCall	Publish call
PublishFBHistEvent	Publish event
PublishFBHistLitReq	Publish literature request
PublishFBHistMsg	Publish message

PublishFBHistOpTask	Publish op task
PublishFBHistOther	Publish other
PublishFBHistProjTask	Publish project task
PublishFBHistPubEvent	Publish public event
PublishFBHistSales	Publish sales
PublishFBHistToDo	Publish todo
PublishFBFilterByDate	Dates to publish 0 - today 1 - yesterday 2 - tomorrow 3 - this week 4 - last week 5 - next week 6 this month 7 last month 8 next month 9 - this year 10 - next year 11 - date range
PublishFBStartDate	The start date of the range
PublishFBEndDate	The end date of the range
PublishFBFreq	Frequency in minutes

WRITECALENDARPREFS RETURN VALUES

WriteCalendarPrefs Return Values

Value	Description
1	Success
0	No container passed
-1	Not a master rights user or invalid user name
-2	User ini file doesn't exist
-3	Cannot open the cal table

Reading Personal Preferences

The ReadPersonalPrefs function gets the personal preferences for the passed or current user.

READPERSONALPREFS Input NV pairs

ReadPersonalPrefs Input NV Pairs

Name	Description
UserName	User name passed

READPERSONALPREFS OUTPUT NV pairs

ReadPersonalPrefs Output NV Pairs

Name	Description
UserName	User name passed
Title	The user's title
Dept	The user's department
Phone	The user's phone number
Fax	The user's fax

READPERSONALPREFS RETURN CODES

ReadPersonalPrefs Return Codes

Value	Description
1	Success
0	No container passed
-1	Not a master rights user or invalid user name
-2	User ini file doesn't exist

Updating Personal Preferences

The WritePersonalPrefs function updates the personal preferences for the passed or current user.

WRITEPERSONALPREFS Input NV pairs

WritePersonalPrefs Input NV Pairs

Name	Description
UserName	User name passed

WRITEPERSONALPREFS OUTPUT NV pairs

WritePersonalPrefs Output NV Pairs

Name	Description
UserName	User name passed
Title	the user's title
Dept	The user's department
Phone	The user's phone number
Fax	The user's fax

WRITEPERSONALPREFS RETURN CODES

WritePersonalPrefs Return Codes

Value	Description
1	Success
0	No container passed
-1	Not a master rights user or invalid user name
-2	User ini file doesn't exist

Reading Record Preferences

The ReadRecordPrefs function gets the record preferences for the passed or current user.

READRECORDPREFS Input NV pairs

ReadRecordPrefs Input NV Pairs

Name	Description
UserName	User name passed

READRECORDPREFS OUTPUT NV pairs

ReadRecordPrefs Output NV Pairs

Name	Description
UserName	User name passed
UseContactForTitle	Use contact instead of company in title – 1 = cont, 0 company
SelectFieldContents	When a field gets focus select its contents
AutoOpenOrgTree	Open org tree when record object is maximized

ShowDatesInWords	Show user-defined dates in words
DateFormat	0 = MMM d, yy 1 = MMMM dd, yyyy 2 = d MMM yy 3 = d. MMM yy 4 = dd MMMM yy
RightAlignNumbers	Show numerics right-aligned
ShowSortByFieldInStatus	Show sort-by field on status bar
ZipValidationMode	0= none, 1 primary, 2 show zip dialog
Show9DigitZip	Show 5 or 9 digits in zip code lookup validation window
UseDarkBgd	Use a dark background color on the RO
LargeFont	Use a large font – doesn't affect 640x480 resolution
LabelColor	Windows color for the labels
DataColor	Windows color for the data

READRECORDPREFS RETURN CODES

ReadRecordPrefs Return Codes

Value	Description
1	Success
0	No container passed
-1	Not a master rights user or invalid user name
-2	User ini file doesn't exist

Updating Record Preferences

The WriteRecordPrefs function updates the record preferences for the passed or current user.

WRITERECORDPREFS Input NV pairs

WriteRecordPrefs Input NV Pairs

Name	Description
UserName	User name passed
UseContactForTitle	Use contact instead of company in title – 1 = cont, 0 company

SelectFieldContents	When a field gets focus select its contents
AutoOpenOrgTree	Open org tree when record object is maximized
ShowDatesInWords	Show user-defined dates in words
DateFormat	0 = MMM d, yy 1 = MMMM dd, yyyy 2 = d MMM yy 3 = d. MMM yy 4 = dd MMMM yy
RightAlignNumbers	Show numerics right-aligned
ShowSortByFieldInStatus	Show sort-by field on status bar
ZipValidationMode	0= none, 1 primary, 2 show zip dialog
Show9DigitZip	Show 5 or 9 digits in zip code lookup validation window
UseDarkBgd	Use a dark background color on the RO
LargeFont	Use a large font – doesn't affect 640x480 resolution
LabelColor	Windows color for the labels
DataColor	Windows color for the data

WRITERECORDPREFS RETURN CODES

WriteRecordPrefs Return Codes

Value	Description
1	Success
0	No container passed
-1	Not a master rights user or invalid user name
-2	User ini file doesn't exist

Reading Schedule Preferences

The ReadSchedulePrefs function gets the schedule preferences for the passed or current user.

READSCHEDULEPREFS Input NV pairs

ReadSchedulePrefs Input NV Pairs

Name	Description
UserName	User name passed

READSCHEDULEPREFS OUTPUT NV pairs

ReadSchedulePrefs Output NV Pairs

Name	Description
UserName	User name passed
ConflictOn	Check for timing conflicts when scheduling
CarryCompletionNotesOnFollowUp	Carry over completion notes when scheduling follow ups
StartTimerOnComplete	Start timer when completing activities
ShowDetailsInActivityListingWindow	Show the details section in activity listing window
SyncContactWithActivityListingWindow	Sync the contact window with the activity listing window
WarnAboutCompleteMultiLinkActiv	Show alert when completing an activity with others associated.
WarnAboutEditMultiLinkActiv	Show alert when editing an activity with others associated
WarnAboutDeleteMultiLinkActiv	Show alert when deleting an activity with others associated

READSCHEDULEPREFS RETURN CODES

ReadSchedulePrefs Return Codes

Value	Description
1	Success
0	No container passed
-1	Not a master rights user or invalid user name
-2	User ini file doesn't exist

Updating Schedule Preferences

The WriteSchedulePrefs function updates the record preferences for the passed or current user.

WRITESCHEDULEPREFS Input NV pairs

WriteSchedulePrefs Input NV Pairs

Name	Description
UserName	User name passed
ConflictOn	Check for timing conflicts when scheduling
CarryCompletionNotesOnFollowUp	Carry over completion notes when scheduling follow ups

StartTimerOnComplete	Start timer when completing activities
ShowDetailsInActivityListingWindow	How the details section in activity listing window
SyncContactWithActivityListingWindow	Sync the contact window with the activity listing window
WarnAboutCompleteMultiLinkActiv	Show alert when completing an activity with others associated.
WarnAboutEditMultiLinkActiv	Show alert when editing an activity with others associated
WarnAboutDeleteMultiLinkActiv	Show alert when deleting an activity with others associated

WRITESCHEDULEPREFS RETURN CODES

WriteSchedulePrefs Return Codes

Value	Description
1	Success
0	No container passed
-1	Not a master rights user or invalid user name
-2	User ini file doesn't exist

Reading Alarm Preferences

The ReadAlarmPrefs function gets the alarm preferences for the passed or current user.

READALARMPREFS Input NV pairs

ReadAlarmPrefs Input NV Pairs

Name	Description
UserName	User name passed

READALARMPREFS OUTPUT NV pairs

ReadAlarmPrefs Output NV Pairs

Name	Description
UserName	User name passed
AlarmType	0 = none, 1 – pop up, 2 – taskbar notifications
AlarmsLead	Time before an event that an alarm fires
AlarmFreq	Scan for alarm every xx seconds

TaskBarReminder	Reminder shown for x minutes
IgnoreSnooze	Amount of to snooze an ignored alarm
PageAlarm	Page user with alarm when not acknowleged within xx minutes.
GMAlarmSound	Path to the alarm sound

READALARMPREFS RETURN CODES

ReadAlarmPrefs Return Codes

Value	Description
1	Success
0	No container passed
-1	Not a master rights user or invalid user name
-2	User ini file doesn't exist

Updating Alarm Preferences

The WriteAlarmPrefs function updates the alarm preferences for the passed or current user.

WRITEALARMPREFS Input NV pairs

ReadAlarmPrefs Input NV Pairs

Name	Description
UserName	User name passed
AlarmType	0 = none, 1 – pop up, 2 – taskbar notifications
AlarmsLead	Time before an event that an alarm fires
AlarmFreq	Scan for alarm every xx seconds
TaskBarReminder	Reminder shown for x minutes
IgnoreSnooze	Amount of to snooze an ignored alarm
PageAlarm	Page user with alarm when not acknowleged within xx minutes.
GMAlarmSound	Path to the alarm sound

WRITEALARMPREFS RETURN CODES

WriteAlarmPrefs Return Codes

Value	Description
1	Success
0	No container passed
-1	Not a master rights user or invalid user name
-2	User ini file doesn't exist

Reading Lookup Preferences

The ReadLookupPrefs function gets the lookup preferences for the passed or current user.

READLOOKUPPREFS Input NV pairs

ReadLookupPrefs Input NV Pairs

Name	Description
UserName	User name passed

READLOOKUPPREFS OUTPUT NV pairs

ReadLookupPrefs Output NV Pairs

Name	Description
UserName	User name passed
SyncContact	Sync the contact window with the search center window
InShrunkenMode	Appear in shrunken mode when finding by
SyncDelay	Lookup alignment delay when typing in tenths of a second
DefField	Default lookup field 0 – contact, 1 = company
SelectAction	When a rec is selected in search cente 0 = move the search center window to the back 1 = close the search center window 2 = minimize the search center windowr

READLOOKUPPREFS RETURN CODES

ReadLookupPrefs Return Codes

Value	Description
1	Success
0	No container passed

-1	Not a master rights user or invalid user name
-2	User ini file doesn't exist

Updating Alarm Preferences

The WriteLookupPrefs function updates the lookup preferences for the passed or current user.

WRITELOOKUPPREFS Input NV pairs

WriteLookupPrefs Input NV Pairs

Name	Description
UserName	User name passed
SyncContact	Sync the contact window with the search center window
InShrunkenMode	Appear in shrunken mode when finding by
SyncDelay	Lookup alignment delay when typing in tenths of a second
DefField	Default lookup field 0 – contact, 1 = company
SelectAction	When a rec is selected in search cente 0 = move the search center window to the back 1 = close the search center window 2 = minimize the search center windowr

WRITELOOKUPPREFS Return Codes

WriteLookupPrefs Return Codes

Value	Description
1	Success
0	no container passed
-1	Not a master rights user or invalid user name
-2	user ini file doesn't exist

Reading Pager Preferences

The ReadPagerPrefs function gets the pager preferences for the passed or current user.

READPAGERPREFS Input NV pairs

ReadPagerPrefs Input NV Pairs

Name	Description
UserName	User name passed

READPAGERPREFS OUTPUT NV pairs

ReadPagerPrefs Output NV Pairs

Name	Description
UserName	User name passed
GoldPageInstalled	Is the goldpage application installed?
Terminal	Terminal pager number
PIN	The pin for the pager
MaxChars	The number of max chars for a pager
PagerEmail	Email page address

READPAGERPREFS Return Codes

ReadPagerPrefs Return Codes

Value	Description
1	Success
0	No container passed
-1	Not a master rights user or invalid user name
-2	User ini file doesn't exist

Updating Pager Preferences

The WritePagerPrefs function updates the pager preferences for the passed or current user.

WRITEPAGERPREFS Input NV pairs

WritePagerPrefs Output NV Pairs

Name	Description
UserName	User name passed
GoldPageInstalled	Is the goldpage application installed?
Terminal	Terminal pager number

PIN	The pin for the pager
MaxChars	The number of max chars for a pager
PagerEmail	Email page address

WRITEPAGERPREFS Return Codes

WritePagerPrefs Return Codes

Value	Description
1	Success
0	No container passed
-1	Not a master rights user or invalid user name
-2	User ini file doesn't exist

Reading Miscellaneous Preferences

The ReadMiscPrefs function gets the miscellaneous preferences for the passed or current user.

READMISCPREFS Input NV pairs

ReadMiscPrefs Input NV Pairs

Name	Description
UserName	User name passed

READMISCPREFS OUTPUT NV pairs

ReadMiscPrefs Output NV Pairs

Name	Description
ShowWhatsNew	Show whats new in the info center when logging in
TimeIn24Hr	Show time in 24/military style
DateInLocalFormat	Show dates in local format
ShowPageStatus	Show status while paging
OldMenu	Use the old GM4 style menu
EPOCH	The EPOCH year
MSMailUser	The MS outlook username if not the same as the GM user name

READMISCPREFS Return Codes

ReadPagerPrefs Return Codes

Value	Description
1	Success
0	No container passed
-1	Not a master rights user or invalid user name
-2	User ini file doesn't exist

Updating Miscellaneous Preferences

The WriteMiscPrefs function updates the miscellaneous preferences for the passed or current user.

WRITEMISCPREFS Input NV pairs

WriteMiscPrefs Input NV Pairs

Name	Description
ShowWhatsNew	Show whats new in the info center when logging in
TimeIn24Hr	Show time in 24/military style
DateInLocalFormat	Show dates in local format
ShowPageStatus	Show status while paging
OldMenu	Use the old GM4 style menu
EPOCH	The EPOCH year
MSMailUser	The MS outlook username if not the same as the GM user name

WRITEMISCPREFS Return Codes

WriteMiscPrefs Return Codes

Value	Description
1	Success
0	No container passed
-1	Not a master rights user or invalid user name
-2	User ini file doesn't exist

Reading the Database Engine Type (7.0 or higher)

The GetDbEngineType function gets the database engine type based on a passed table name.

GETDBENGINETYPE Input NV pairs

GetDbEngineType Input NV Pairs

Name	Description
Table	The table name you are trying to open - if not passed, assumed to be CONTACT1

GETDBENGINETYPE Return Codes

GetDbEngineType Return Codes

Value	Description
0	No container passed
-1	Table name not passed
-2	Table name invalid
-3	Could not open table
1	Table is MSSQL
2	Table is Firebird
3 or higher	Unknown DB type

Reading a List of GoldMine User Groups

The GetGMUserGroups function returns a list of GoldMine user groups and their users.

GETGMUSERGROUPS OUTput NV pairs

GetGMUserGroups Output NV Pairs

Name	Description
GROUP	NV container for EACH group containing: GroupNumber – the group's internal number Name – the name of the group UserCount – the number of users in the group UserList – a list of the users in the group delimited by ;

GETGMUSERGROUPS Return Codes

GetGMUserGroups Return Codes

Value	Description
1	Success
0	No container passed
-1	Could not open data tables

Creating or Updating GoldMine User Groups

The WriteGMUserGroup function creates or updates a GoldMine user group.

WRITEGMUSERGROUP Input NV pairs

WriteGMUserGroup Input NV Pairs

Name	Description
Name	The name of the group to update or create
RecID	The record number of the group if updating

WRITEGMUSERGROUP Return Codes

WriteGMUserGroup Return Codes

Value	Description
0	No container passed
-1	No group name
-2	Could not write data
-3	Not a master user
-4	Could not lock record
1	Success

Adding a GoldMine User to a Group

The AddGMGroupUser function adds a GoldMine user to a group.

ADDGMGROUPUSER Input NV pairs

AddGMGroupUser Input NV Pairs

Name	Description
UserName	The name of the user to add to the group

GroupName	The group name or the group number to add the user to

ADDGMGROUPUSER Return Codes

AddGMGroupUser Return Codes

Value	Description
0	No container passed
-1	No name or group passed
-2	Could not open users table
-3	Could not lock user record
-4	Could not find user record
-5	Invalid group passed
-6	Not a master user
1	Success or user already group member

Removing a GoldMine User from a Group

The RemoveGMGroupUser function removes a GoldMine user from a group.

REMOVEGMGROUPUSER Input NV pairs

RemoveGMGroupUser Input NV Pairs

Name	Description
UserName	The name of the user to remove from the group
GroupName	The group name or the group number to remove the user from

REMOVEGMGROUPUSER Return Codes

RemoveGMGroupUser Return Codes

Value	Description
0	No container passed
-1	No name or group passed
-2	Could not open users table
-3	Could not lock user record

-4	Could not find user record
-5	Invalid group passed
-6	Not a master user
1	Success or user already group member

Creating or Updating an Opportunity or Project

The WriteOpProj function updates an opportunity or project.

WRITEOPPROJ Input NV pairs

In addition to the following, the user can pass the custom user defined fields (GM 6.6 or higher) that they have created.

WriteOpProj Input NV Pairs

Name	Description
RecID	If the item is an update – the recid of the item to update
OpID	The opportunity rec id to attach to
RecType	O or P
AccountNo	The contact to attach to's account no
User	The gm user to assign the item to
Flags	Flags for the item
Company	The company this item involves
Contact	The contact the item involves
Name	Name of the item
Status	The status of the item
Cycle	The cycle of the item
Stage	The item's stage
Source	The item's source
F1	The F1 value
F2 or CompRecID	The rec id of the company from Company field
F3 or Units	The number of units this item involves

StartDate	The start date
ClosedDate	The date closed
CloseBy	The date to close by
ForProb	The probability of the item success
ForAmt	The projected value of the item
CloseAmt	The actual value of the item
Notes	Item notes

WRITEOPPROJ Return Codes

WriteOpProj Return Codes

Value	Description
1	Success
0	No container passed

Working with GoldMine Plug-ins

Overview

This chapter contains information geared toward individuals with at least an intermediate knowledge of programming.

GoldMine 7.0 supports integrations based on ActiveX controls or HTML. To use either of these integration methods, you must first create an ActiveX control or an HTML file or web site to integrate with.

Using ActiveX Plug-in Support

The ActiveX structure allows the most control and can be made with almost any language, including C++, Delphi, VB and the .NET languages. When used in conjunction with the other GoldMine APIs, Active X is extremely powerful.

Within the ActiveX support, there are 5 methods that can be implemented in your control to allow for stronger interaction with GoldMine. These functions are not necessary to implement:

```
public void GMOnStart(long hWnd)
```

This is the only function that passes a parameter. The parameter is the HWND (window handle) of the container window in GoldMine. You can then use the Windows API SendMessage() call to control what happens to the container. This is for situations where you want to implement a Close button, since the control is late bound in GoldMine, and cannot expose events.

```
public void GMOnActivate()
```

This function will tell you when the user has given your control's container focus in GoldMine.

```
public void GMLostFocus()
```

Called whenever the user gives focus to another object when your control had focus.

```
public void GMOnDestruct()
```

Called when the window is just about to close. This allows you the opportunity to clean up.

```
public void GMHandleFile(BSTR sPath)
```

Used to open associated files with your plug-in. the passed Path is the path to the file itself that your plug-in described it could handle.

Using HTML Plug-in Support

HTML plug-in support also has great potential. The HTML will attempt to call a JavaScript or VBScript function named like the last 3 ActiveX methods, with exactly the same capabilities:

```
GMOnActivate()
GMLostFocus()
GMOnDestruct()
```

The GMOnStart() function is not supported in HTML.

Plug-In Description File

The plug-in description file is a well formed XML file that describes the plug-in. The extension for the file is .GME (for GoldMine Extension).

HTML Plug-in Description File

The following example shows the structure for the HTML plug-in.

```
<PlugInDefs>
<PlugInDef>
<URL>http://gmail.google.com/gmail</URL>
<QueryString>q=&lt;&lt;&amp;Address1&gt;&gt;, &lt;&lt;&amp;City&gt;&gt;,
&lt;&lt;&amp;State&gt;&gt;, &lt;&lt;&amp;Zip&gt;&gt;</QueryString>
 <Description>
<Language Locale="1033" IsDefault="1">
<Name>G-Mail</Name>
<Publisher>Google</Publisher>
<Description>Launches Google's Gmail Service</Description>
<Menu>Launch GMAIL</Menu>
 <MenuPath>Web Based Tools\\Google</MenuPath >
</Language>
<Language Locale="4000">
<Name>eegay ale-may</Name>
<Publisher>oogle-Gay</Publisher>
<Description>aunches-Lay oogle-Gay’s eegay ale-may Urvice-Say</Description>
<Menu>aunch-Lay eegay ale-may</Menu>
 <MenuPath>eb-Way ased-Bay ools-Tay\\oogle-Gay</MenuPath >
</Language>
</Description>
<OnDemand>1</OnDemand>
<Startup>1</Startup>
<MultipleInstance>0</MultipleInstance>
<Modal>0</Modal>
<DefaultPos>
<top>50</top>
<left>50</left>
</DefaultPos>
```

```
<DefaultSize>
<width>800</width>
<height>600</height>
</DefaultSize>
<Visible>1</Visible>
<IconFile>google.ico</IconFile>
<InternalName>GOOGLE_MAIL</InternalName>
</PlugInDef>
</PlugInDefs>
```

The root node must be PlugInDefs, and as the name implies, multiple plug-ins can be installed under one definition file. For each plug-in, there is one PlugInDef. The child nodes for PlugInDef are:

Node	Description
<URL>	The URI for the html or site – must be http://, https:// or file://
<QueryString>	The querystring to be tacked on to the end of the URL. Can contain GoldMine field macros that will be evaluated on launch of the plug-in. The macro wrapping structure is <<field>>, like <<&Contact>> or <<Contact1->AccountNo>>. Please note that you must XMLEncode the macros like above.
<Description>	These values describe the item to the user.
<Language>	Uses the locale code associated with the target language. One Langauge structure must be marked as IsDefault, and this one is used in case the target language is not supported by the plug-in. Always use XML entities in place of extended characters. (Ñ would be Ñ)
<Name>	The dialog name and used for security
<Publisher>	Your company name – creates a sub menu under the Plug-ins menu if MenuPath not passed
<Description>	Used in the Help->About Plug-ins button (not there yet)
<Menu>	The text that the user sees for a menu item.
<MenuPath>	Creates a hierarchical set of menus, with each submenu delimited by "\\" – double backslashes
<OnDemand>	Determines if the plug-in is added to the plug-ins menu. 1 = True, 0 = False. If false – then the item is started up with GoldMine.
<StartUp>	Determines if the item is started up with GoldMine. This is for situations where you want it to come up – but if the user closes the window – you want them to be able to access the plug-in via a menu. 1 = startup with GoldMine, 0= don't start with GoldMine.

<MultipleInstance>	Determines if multiple instances of the plug-in are allowed. 1 = allow multiple instances, 0 = false. If false, if the user chooses the menu item for that plug-in – then GoldMine will bring that window to the front and give it focus. non-OnDemand, Modal and non-visible plug-ins are automatically single instance.
<Modal>	Determines if any action can occur outside of the window in GoldMine. 1= Modal, 0 = Modeless. Startup/non-OnDemand items cannot be modal. Modal items are strictly single instance.
<DefaultPos>	Describes the coordinates where your dialog will first show up. This is only used the first time the plug-in is run, and is ignored for Modal plug-ins, which are automatically centered in relation to the GoldMine window.
<top>	Number of pixels from the top of the screen.
<left>	Number of pixels from the left of the screen.
<DefaultSize>	describes the height and width of the dialog for first time use, or for modal windows – which cannot be resized.
<width>	Width of the window in pixels.
<height>	Height of the window in pixels.
<Visible>	Determines if the user can see the window. Not recommended for HTML based plug-ins.
<IconFile>	If you have an ico file that you want the item to use, then put it in the plug-ins folder and specify it here.
<InternalName>	This is a name that you give to your plug-in that can then be used in the INI files to block/grant access. If it is not passed it will be made up of a concatenation of the Publisher name and the Name fields for the default locale, using only the following characters: “ABCDEFGHIJKLMNOPQRSTUVWXYZ_1234567890”

ActiveX Plug-in Description File

The following example shows the structure for the ActiveX plug-in.

```
<PlugInDefs>
<PlugInDef>
<ProgID>myApp.ClassInstance</ProgID>
<Installer>myAppInstaller.exe</Installer>
 <Description>
<Language Locale="1033" IsDefault="1">
<Name>My Fantastical App</Name>
<Publisher>JCS</Publisher>
<Description>This app does it all!!!</Description>
<Menu>The most amazing app EVER</Menu>
 <MenuPath>You\\Can\\Expect\\To Be\\AMAZED</MenuPath >
```

```
</Language>
<Language Locale="4000">
<Name>eegay ale-may</Name>
<Publisher>oogle-Gay</Publisher>
<Description>aunches-Lay oogle-Gay's eegay ale-may Urvice-Say</Description>
<Menu>aunch-Lay eegay ale-may</Menu>
<MenuPath> ou-Yay\\an-Kay\\Expect-ay\\o-tay ebay\\AMAZED-AY</MenuPath >
</Language>
</Description>
<OnDemand>1</OnDemand>
<Startup>1</Startup>
<MultipleInstance>0</MultipleInstance>
<Modal>0</Modal>
<DefaultPos>
<top>50</top>
<left>50</left>
</DefaultPos>
<DefaultSize>
<width>800</width>
<height>600</height>
</DefaultSize>
<Visible>1</Visible>
<IconFile>MYAPP.ico</IconFile>
<InternalName>BEST_APP_EVER</InternalName>
<HandledFileExtensions>doc;xls;pdf;txt;ini</HandledFileExtensions>
<Methods>
<Method>
<Language Locale="1033" IsDefault="1">
<Menu>Launch The app</Menu>
</Language>
</Method>
<Method call="Configure">
<Language Locale="1033" IsDefault="1">
<Menu>Configure the bliss</Menu>
</Language>
<Language Locale="4000">
<Menu>Onfigure-Kay ah-they iss-blay</Menu>
</Language>
</Method>
</Methods>
</PlugInDef>
</PlugInDefs>
```


Although it is very similar to the HTML plug-in description, there are 2 primary differences: the ProgID and Installer nodes instead of the URL and QueryString nodes.

The ProgID is the ProgID for your ActiveX control, and the Installer is the installer name for the application. The Installer should be located in a folder named Installers under the plug-in directory.

There is also the "HandledFileExtensions" element that can be added to handle files of certain extensions with your plug-in internally in GoldMine. This means that if there is a linked document, email attachment, or other internally attached file that would normally launch a third party application, the path to the file will be passed to your plug-in via the GMHandleFile call. This does not mean external to GoldMine that opening that file will launch GoldMine and your plug-in. However, it should be a simple task to write an .exe wrapper for your plug-in (since its ActiveX based, after all) and associate the file types to that exe wrapper.

The Methods Section allows you to call custom methods in your application. When in use the Description's Menu node becomes a sub-menu with all of the methods that you have described. A method is described by the Method node with an optional attribute "call" which tells GoldMine what internal method to call. The internal method must be public and expect no parameters. It must also return nothing (void or sub). The language portion works exactly like the description node's does – except it only has the Menu entry.

Security and Plug-in Directories

Using `GM.INI` or the `User.INI`, a user/admin can block the use of plug-ins altogether, block individual plug-ins and also add user specific directory for more plug-ins.

Security

For security, `GM.INI` has precedence over the user INI file. There are two methods – Optimistic and Pessimistic. You can have different methods for `GM.INI` and the user INI, but Pessimistic will win out.

The Optimistic method is as follows:

```
[PlugIns]
allow_by_default=1
```

The Pessimistic method is as follows:

```
[PlugIns]
deny_by_default=1
```

If you had allow_by_default=0, then this would be the same as deny_by_default=1 – and vice versa. If the keys are missing, then the method is assumed to be Optimistic.

If you are using the Optimistic method, then you do not have to add anything besides blocked plug-ins to the INI files. If you are using the Pessimistic method, then you must give a plug-in permission to run.

For example, if you have a plug-in with a Name node of "Evil Plugin …"

The INI name for this would be EVILPLUGIN unless you added the InternalName element to your plug-in description.

To block the plug-in with Optimistic mode:

```
[PlugIns]
allow_by_default=1 or deny_by_default=0
EVILPLUGIN=0
```

To allow a plug-in with Pessimistic mode:

```
[PlugIns]
deny_by_default=1 or allow_by_default=0
```

```
GOODPLUGIN=1
```

Adding a Local Plug-in Directory

By default – the plug-in directory is under %SysDir%/Plug-ins and in server installs this means that all users will have the plug-ins under that folder. If a user wanted to add his own local plug-in directory – he could add it to his user INI:

```
[PlugIns]
LocalPath=c:\personal\GMPlugIns
```

The user will still get the global level programs (assuming they're not blocked) – so make sure there's no duplication between the two.

Sample Plug-ins

The following are examples of the GoldMine plug-in capabilities

gmail.gme

This plug-in opens a browser window to the Google mail address. It demonstrates the basic capability of opening a browser window from GoldMine.

```
<?xml version="1.0" encoding="UTF-8"?>
<PlugInDefs>
<PlugInDef>
<URL>http://gmail.google.com/gmail</URL>
<Description>
<Language Locale="1033" IsDefault="1">
<Name>G-Mail</Name>
<Publisher>Google</Publisher>
<Description>Launches Google's Gmail Service</Description>
<Menu>Launch GMAIL</Menu>
</Language>
</Description>
<OnDemand>1</OnDemand>
<Startup>1</Startup>
<MultipleInstance>0</MultipleInstance>
<Modal>0</Modal>
<DefaultPos>
<top>50</top>
<left>50</left>
</DefaultPos>
<DefaultSize>
<width>800</width>
<height>600</height>
</DefaultSize>
<Visible>1</Visible>
</PlugInDef>
</PlugInDefs>
```

External.gme

This plug-in allows a user to store more than the 254 custom fields for a contact record externally. Users can select any contact record, then select the plug in, to either add new information or update existing information depending on what is found in the database.

```
<?xml version="1.0" encoding="UTF-8"?>
<PlugInDefs>
<PlugInDef>
<URL>http://localhost/gmplus.asp</URL>
<QueryString>accountno=&lt;&lt;&amp;Accountno&gt;&gt;</QueryString>
<Description>
<Language Locale="1033" IsDefault="1">
<Name>Extra Fields</Name>
<Publisher>Robie</Publisher>
<Description>Access External Tables</Description>
<Menu>Access External Tables</Menu>
</Language>
</Description>
<OnDemand>1</OnDemand>
<MultipleInstance>1</MultipleInstance>
<Modal>0</Modal>
<DefaultPos>
<top>50</top>
<left>50</left>
</DefaultPos>
<DefaultSize>
<width>600</width>
<height>590</height>
</DefaultSize>
<Visible>1</Visible>
</PlugInDef>
</PlugInDefs>
```

gmplus.asp

Following is the source listing for gmplus.asp, which is the corresponding ASP page for the External.gme plug-in.

NOTE: The following code sample uses text wrapping in order to fit the sample on these pages. Make sure that the lines in your actual code do not wrap.

```
<html>
<body>
<h3>External Location Information</h3>
<%
Dim action
Dim DSNConnection
Dim SQLTable
'Update the DSN information here to access the SQL database HERE.
```

```
DSNConnection = "Driver=SQL
Server;Server=CompanyServerName;Database=GMplus;Uid=sa;Pwd=sa;"
'Update to table in database
SQLTable = "GoldPlus"

'add/edit additional fields here
Dim strdocument, strlocation, strextrastuff1, straccountno
'add/edit additional fields here too
strdocument = Replace(Request("document"), "'", "''")
strlocation = Replace(Request("location"), "'", "''")
strextrastuff1 = Replace(Request("extrastuff1"), "'", "''")
straccountno = Replace(Request("accountno"), "'", "''")

'This section updates fields if the accountno is found in the database
if Request("action")="update" then

set conn=Server.CreateObject("ADODB.Connection")
conn.Open (DSNConnection)

'This is the SQL statement that updates information, so you will need to
add/edit fields here too.
set rs = Server.CreateObject("ADODB.recordset")
strSQL = "UPDATE "+ SQLTable +" SET document = '" + strdocument +"',
location = '" + strlocation + "', extrastuff1 = '" + strextrastuff1 + "'
WHERE accountno = '" + straccountno + "'"
Conn.Execute (strSQL)

conn.close
set conn = nothing
set strSQL = nothing
'This does a redirect to the update page once the data is entered into the
SQL database
Response.write("<meta http-equiv=refresh
content=0;url=gmplus.asp?accountno=" + straccountno + ">")

'*****************************************************************************
*************************************

'This section does the addition of the fields if they are not found in the
database
else if Request("action")="add" then

set conn=Server.CreateObject("ADODB.Connection")
conn.Open (DSNConnection)

'This adds new information if it is not found in the database
set rs = Server.CreateObject("ADODB.recordset")
strSQL = "INSERT INTO "+ SQLTable +"
(accountno,document,location,extrastuff1) VALUES ('" + straccountno + "','"
+ strdocument + "','" + strlocation + "','" + strextrastuff1 + "')"
Conn.Execute (strSQL)
```

```
conn.close
set conn = nothing
set strSQL = nothing
'This does a redirect to the update page once the data is entered into the
SQL database.
Response.AddHeader "Location", "/gmplus.asp?accountno='" + straccountno +
"'"
end if

set conn=Server.CreateObject("ADODB.Connection")
conn.Open (DSNConnection)

set rs = Server.CreateObject("ADODB.recordset")
rs.Open "SELECT accountno, document, location, extrastuff1 from "+ SQLTable
+" where accountno ='"+ straccountno +"'" , conn

'*************************************************************************
**************
'if the AccountNo is NOT found, display the ADD form
if rs.eof AND rs.bof then
%>
<form action="gmplus.asp" method="get">
<input type="hidden" name="action" value="add">
<% Response.Write("<input type=hidden name=accountno value="+ straccountno
+">")%>
<table border="1">
<tr>
<td>Document</td><td><input type="text" name="document" size="30"></td>
<tr>
</tr>
<td>Location</td><td><input type="text" name="location" size="30"></td>
<tr>
</tr>
<td>Extra Stuff 1</td><td><input type="text" name="extrastuff1"
size="30"></td>
</tr>
</table>
<input type="Submit" value="add">
</form>

<%'*************************************************************************
*****************
else
'if the AccountNo IS found, display the UPDATE form
%>
<form action="gmplus.asp" method="get">
<input type="hidden" name="action" value="update">
<% Response.Write("<input type=hidden name=accountno value="+ straccountno
+">")%>
<table border="1">
<tr>
```

```
<td>Document</td><td><input type="text" name="document" value="<%= rs
("document") %>" size="30"></td>
</tr>
<tr>
<td>Location</td><td><input type="text" name="location" value="<%= rs
("location") %>" size="30"></td>
</tr>
<tr>
<td>Extra Stuff 1</td><td><input type="text" name="extrastuff1" value="<%=
rs("extrastuff1") %>" size="30"></td>
</tr>
</table>
<input type="Submit" value="update">
</form>

<%'*************************************************************************
****************
end if
end if
%>
</body>
</html
```

Using Xbase Expressions

Overview

This chapter contains information geared toward individuals with at least an intermediate knowledge of programming.

IMPORTANT: Improper use of these functions may result in data that is not recoverable. Be sure to back up your data frequently.

> **TIP:** For details on data backups, see "Backing up Data" in Maintaining GoldMine.

GoldMine offers a variety of Xbase expression functions to:

- Manipulate data for comparison, such as for creating filters and groups.
- Store data, such as for global replacements and updates to field data (LOOKUP.INI).
- Evaluate and return data when using DDE and GMXS32.DLL function calls.

To ensure that your Xbase functions work correctly, GoldMine also features a real-time expression tester. To activate the tester on an active record window, press Ctrl-Shift-D.

> **TIP:** Xbase functions are also known as dBASE functions.

Filter expressions work equally well on Xbase or SQL tables. With SQL, the Xbase filter is evaluated on the client side, not the server side.

The following pages list Xbase functions in three sections:

- Function/Parameter Types
- Conditionals, Operators, and Logical Evaluators
- Xbase Functions

Function/Parameter Types

Xbase functions recognize and return several types of data. These data types represent the format of the data, such as a number. To properly evaluate and return a value, a function must include the correct parameter types. For example, a function may require that a date be passed as a parameter. Trying to pass a name to the function would not be accepted. In many cases, you can use a special function to convert one data type to another.

Data types may be referenced literally, either as a field name of a specific type, or as the result of an Xbase function.

The following list describes valid data types for Xbase functions and shows examples of use when referenced as a literal, field value, or function result.

String	Sequence of any printable character. Literal use: "my string" Field use: Upper(Contact1->Company) Function Use: Upper(Substr("test123",5,3))
Date	Special numeric value representing a date. Literal use: {03/10/1999} Field use: DTOS(Contact2->UBirthday) Function use: DTOS(DATE())
Numeric	Value representing a number. Literal use: 100 Field use: STR(Contact2->UBalance) Function use: STR(100 + VAL("100"))
Boolean	Value that results whenever a comparison is made. Boolean values are either TRUE or FALSE.

For an expanded description of Boolean expressions, see “Using Boolean Expressions” in the Online Help.

Conditionals, Operators, and Logical Evaluators

A function can manipulate values by using one of the following:

- *Conditional*: Compares one value to another, using the specified standard or condition, such as “equal to,” “greater than,” and so on.
- *Operator*: Performs an arithmetic operation on the values, such as addition or multiplication.
- *Logical* evaluator: Compares values as a true/false condition, so that a value either meets or fails the standard for selection. This type of comparison is also known as a Boolean operator.

You can use the following conditionals, operators, and logical evaluators in conjunction with the Xbase functions.

Conditionals

Conditional:	>	
Description:	Greater than	
Applies to:	All types	
Examples:	1>2	returns: FALSE
	"BBC">"ABC"	returns: TRUE
	Date()>Date()-10	returns: TRUE

Conditional:	<	
Description:	Less than	
Applies to:	All types	
Examples:	300<400	returns: TRUE
	"MARCELA"<"NELSON"	returns: TRUE
	Date() < Date()-7	returns: FALSE

Conditional:	<>	
Description:	Greater/Less than (not equal)	
Applies to:	All types	
Examples:	250<>2500	returns: TRUE
	"ABC"<>UPPER("abc")	returns: FALSE
	Date()<>Date()+3	returns: TRUE

Conditional:	>=	
Description:	Greater than or Equal to	
Applies to:	All types	
Examples:	100>=99	returns: TRUE
	"ABC">="BBC"	returns: FALSE
	Date()+10>=-Date()	returns: TRUE

Conditional:	<=	
Description:	Less than or equal to	
Applies to:	All types	
Examples:	100<=99	returns: FALSE
	"ABC"<="BBC”	returns: TRUE
	Date()+10<=Date()	returns: FALSE

Operators

Operator:	+	
Description:	Adds one value to another value	
Applies to:	All types	
Examples:	"ABC"+"DEF"	returns: "ABCDEF"
	100+23	returns: 123
	Date()+7	returns: date one week from today

Operator:	-	
Description:	Subtracts one value from another value	
Applies to:	Numeric and Date types	
Examples:	123-100	returns: 23
	Date()-140	returns: date of two weeks ago

Operator:	/	
Description:	Divides one number by another	
Applies to:	Numeric type	
Example:	100/4	returns: 25

Operator:	*	
Description:	Multiplies one value by another	
Applies to:	Numeric type	
Example:	100*5	returns: 500

Operator:	%
Description:	Modulus
Applies to:	Numeric type

Example:	100%33	returns: 1

Logical Evaluators

Logical:	.OR.
Description:	Returns TRUE if either condition is TRUE
Example:	State="CA" .OR. Zip="99999"

Logical:	.AND.
Description:	Returns TRUE only if all conditions are TRUE
Example:	Company="GoldMine, Inc." .AND. Phone1="(310)454-6800"

Logical:	.NOT.
Description:	Returns the opposite of the condition being tested
Example:	.NOT. City="San Francisco”

Xbase Functions

GoldMine recognizes four types of Xbase functions as valid

- *String*: Use primarily for manipulating string data types. A string function can return other data types.
- *Date*: Use for any date-related operations. A date function can return other data types.
- *Numeric*: Use for numeric operations. A numeric function can return other data types.
- *Miscellaneous*: Additional functions that fall outside of the previous three categories of data types. These may return any type of data.

For convenience, functions are listed under these four categories, according to how they are most typically used. For example, under “Date Functions,” you will find those functions that return numeric or string types from dates.

String Functions

ALLTRIM(<string>)	Returns a string value with both leading and trailing spaces from <string>. Return type: String Example "["+ALLTRIM(" This is a test ")+"]" returns [This is a test].
ASC(<char>)	Returns the ASCII decimal value for <char>. Return type: Numeric Example ASC("A") returns 65.
AT(<string1>, <string2>)	Returns the first position of <string1> in <string2>. Return type: String Example AT("a", "once upon a time") returns 11.
CHR(<byte>)	Returns the ASCII character value for <byte>. Return type: String Example CHR(65) returns A.
FMTTIME(<time>)	Returns a character string (hh:mmap format) derived from <time>. Return type: String Example FMTTIME(TIME()) returns 2:28p.
HTTPSTR(<string>)	Returns <string> with all nonletter/number characters replaced with %values. Return type: String Example HTTPSTR("www.Website.com/some dir/") returns www.Website.com%2Fsome%20dir%2F.
IIF(<condition>,<true result>,<false result>)	Returns either <true result> or <false result>, depending on the Boolean evaluation of <condition>. Return type: Logical Example IIF (99 < 100, "Value is Less than 100", "Value is more than 100") returns "Value is Less than 100".

LEFT(<string>, <length>)	Returns the leftmost <length> characters from <string>. Return type: String Example LEFT("Four score and seven",10) returns Four score.
LEN	See LENGTH below.
LENGTH(<string>)	Returns the number of characters in <string>. Return type: Numeric Example LENGTH("This is a test") returns 14.
LOWER(<string>)	Returns <string> in lower-case letters. Return type: String Example LOWER("TEST THIS FUNCTION") returns test this function.
LTRIM(<string>)	Returns <string> with all leftmost spaces removed. Return type: String Example "[" + LTRIM(" This is a test " + "]" returns [This is a test].
LTRIMPAD(<string>, <length>, <fill>)	Returns <string> with leftmost spaces removed and padded to <length> with <fill> character. Return type: String Example "["+LTRIMPAD(" 1341", 10, "0")+"]" returns 0000001341.
MID(<string>, <start>, <length>)	Returns the string of <length> characters starting at position <start> within <string>. Return type: String Example MID("Four score and seven",6,5) returns score.
PAD(<string>, <length>, <fill>, <mode>)	Returns <string> padded to <length> with the <fill> character. <fill> This optional parameter defaults to a space. <mode> can be 0 for right pad (default), 1 for centered, and 2 for left pad. Return type: String Example PAD(“TEST”, 8, “x”, 1) returns xxTESTxx.

PADL(<string>, <length>, <fill>)	Returns <string> padded to <length> with the <fill> character. <fill> This optional parameter defaults to a space. PADL pads from the left. Return type: String Example PADL("TEST", 8, "x") returns xxxxTEST.
PADR(<string>, <length>, <fill>)	Same as PADL, except that PADR pads the string to the right. Return type: String Example PADR("TEST", 8, "x") returns TESTxxxx.
PROPER(<string>)	Returns a string in which the first letter of each word in <string> is capitalized, and the all following letters are lower-case. Return type: String Example PROPER("fighting IRISH") returns Fighting Irish.
RAT(<string1>,string2>)	Returns the last position of <string1> in <string2>. Return type: Numeric Example RAT("t", "this is a test.") returns 14.
RIGHT(<string>, <length>)	Returns the rightmost <length> characters from <string>. Return type: String Example RIGHT("Four score and seven",5) returns seven.
RTRIM(<string>)	Returns <string> with all rightmost spaces removed. Return type: String Example "[" + RTRIM(" This is a test " + "]" returns [This is a test].
STR(<value>,<length>, <decimals>,<fill char>)	Returns the numeric <value> formatted as a string. The <value> parameter is required. All other parameters are optional. The <length> parameter pads the number to the left with spaces or with the <fill char> if specified. Return type: String Example STR(456, 7, 2, "0") returns 0456.00.

STRTRAN(<string1>, <string2>, <string3>)	Returns a string based on <string1> with all occurrences of <string2> translated to <string3>. Return type: String Example STRTRAN("A1B1C1D1", "1", "x") returns AxBxCxDx.
SUBSTR(<string>, <start>, <length>)	Returns the string of <length> characters starting at position <start> within <string>. Return type: String Example SUBSTR("Four score and seven",6,5) returns score.
TRIM(<string>)	See RTRIM.
UPPER(<string>)	Returns the <string> in upper case. Return type: String Example UPPER("this is a test") returns THIS IS A TEST.
WORD(<string>, <pos>)	Returns the <pos> word within <string>. Return type: String Example WORD("this is a test for the WORD function", 4) returns test.

Date Functions

ACCDATE(<string>)	Returns a date value for <string>, where <string> is a valid GoldMine AccountNo. Return type: Date Example ACCDATE(Contact1->ACCOUNTNO) returns 4/20/99.
AGE(<date>)	Returns the age in years since <date>. Return type: Numeric Example AGE(Contact2->UBDATE) returns 32.

CTOD(<string>)	Returns a date value based on <string>. The <string> parameter should be in the format: mm/dd/yy. Return type: Date Example CTOD("4/20/99")+5 returns 4/25/99.
DATE()	Returns today's date in date format. To add/subtract from this value, simply use the number of days in your expression. For example: DATE()+7 will add seven days to today's date. Return type: Date Example Assuming today's date is 4/20/99, DATE()+7 returns 4/27/99.
DAY(<date>)	Returns that day of the month for the specified <date>. Return type: Numeric Example DAY(DATE()) returns 18.
DOBINDAYS(<date>)	Returns the number of days until the month/day in <date>. Return type: Numeric Example DOBINDAYS(STOD("19681024")) returns 232.
DOW(<date>)	Returns the day of the week in numeric format; for example, Sunday = 0, Monday = 1, and so on Return type: Numeric Example DOW(STOD("19990909")) returns 4.
DOY(<date>)	Returns the number of days elapsed from the beginning of the year in <date> to the month/day in <date>. Return type: Numeric Example DOY(Contact2->UDATE) returns 220.
DTOC(<date>)	Returns a character string (MM/DD/YY format) derived from <date>. Return type: String Example DTOC(Contact2->UDATE) returns 10/24/99.

DTOS(<date>)	Returns a character string (YYYYMMDD format) derived from <date>. Return type: String Example DTOS(Contact2->UDATE) returns 19991024.
MONTH(<date>)	Returns that numeric month for the specified <date>. Return type: Numeric example: Example MONTH(Contact2->UDATE) returns 2.
STOD(<string>)	Converts a <string> value into a date value. <string> should be in the format YYYYMMDD. Return type: Date Example STOD("20000121") returns 1/21/2000.
WDATE(<date>, <format>)	Returns the <date> formatted in variety of ways, based on the optional parameter <format>. <format> 0 mm, dd, yy Jan 21, 00 1 ddd, mmm dd, yy Thu, Jan 21, 00 2 mmm dd Jan 21 3 Long date style Thursday, Jan 21, 2000 The Long date style format 3 is taken from the Windows Regional Settings. Return type: String Example WDATE(Contact2->UDATE, 1) returns Thu, Jan 21, 00.
YEAR(<date>)	Returns the numeric year value of <date>. Return type: Numeric Example YEAR(Contact2->UDATE) returns 2000.

Numeric Functions

CEILING(<number>)	Returns the nearest integer that is greater than or equal to the numeric expression. Return type: Numeric Example CEILING(3.1) returns 4.

COUNTER(<string>, <inc>, <start>, <action>)	Returns a sequence of consecutive numbers each time the expression is evaluated. Each of the parameters is described below. <name> This counter must be unique, and can be a maximum of 10 characters. <inc> Each evaluation of the function increments the counter by the <inc> value. <start> and <action> Optional parameters When <action>is 1, the <start> value is used to reset the counter. The counter is deleted when <action>is 2. COUNTER works similarly to the SEQUENCE function. The key difference is that COUNTER stores the count value between GoldMine sessions, and it is shared by all GoldMine users. The COUNTER function updates a database counter, so COUNTER is much slower than SEQUENCE, which updates a memory counter. The SEQUENCE counter is local to the operation, and its count is lost at the end of the operation. GoldMine can track an unlimited number of uniquely named counters. The counter values are stored in the LOOKUP table. Return type: Numeric Example COUNTER("InvoiceNo", 1, 1000) returns 1000.
FLOOR(<number>)	Returns the nearest integer that is less than or equal to the numeric expression Return type: Numeric Example FLOOR(2.8) returns 2.
INT(<number>)	Returns the integer part of a number without rounding. Return type: Numeric Example INT(123.95) returns 123.
RANDOM(<range>)	Returns a random number. <range> can be any number between 1 and 32,761. The returned random number will range between zero and <range>, not including the range limit. If not specified, the <range> parameter defaults to 32,761. You can generate random numbers up to two billion with the expression random(32761) * random(32761). Return type: Numeric Example RANDOM(10) Returns a number between 0–9.

SEQUENCE(<start>, <inc>)	Returns a sequence of consecutive numbers each time the expression is evaluated. When the expression is first evaluated, the <start> parameter starts the counter. Each subsequent evaluation of the function increments the counter by the <inc> value. The SEQUENCE counter is local to the operation, and its count is lost at the end of the operation. Return type: Numeric Example 1 SEQUENCE(1000,10) returns 1010. Example 2 SEQUENCE(1000,10) SEQUENCE(1000,10) returns 1020.
VAL(<string>)	Converts <string> to a numeric value. Return type: Numeric Example VAL("123.45") returns 123.45.

Miscellaneous Functions

RECCOUNT()	Returns the number of records in Contact1. (May be time-consuming on large SQL tables.) Return type: Numeric Example RECCOUNT() returns 35671
RECNO()	Returns the current record number (Xbase) or RecID (SQL) for the active Contact1 record. Return type: Numeric Example RECNO() returns 351.
RECNOCOUNT()	Returns the current record number and total records. This function is not available for SQL tables. Return type: String Example RECNOCOUNT() returns 236 of 2204.

TIME()	Returns the current time. Return type: Time Example TIME() returns 14:56:22.

Xbase Database Structures

Overview

This chapter is provided for programmers who want to integrate their programs with GoldMine Xbase format database structures.

Third-party developers are encouraged to integrate their products with GoldMine, thereby enhancing both products. If you design a commercial program that works with GoldMine, please contact GoldMine Inc. so we can include your program in our Enhancement Guide.

This chapter describes the file organization and structures of GoldMine databases in an Xbase format. Each database file is listed separately and includes its associated index files, database structure, and special notes. For information about working with GoldMine databases in an SQL format, see SQL Database Structures. The following pages describe the database structures of most GoldMine .DBF files. This chapter does not include a discussion of every database. Security and system database files are not included in this section. You should not interface with these files. For an in-depth discussion on interfacing with GoldMine, visit the Public.GoldMine.Programming newsgroup, which you can access directly from http://www.goldmine.com/.

Most GoldMine files are stored in the GOLDMINE\GMBASE directory. These files include most database and index files. The contact sets (CONT*.*) are stored in a separate directory to allow GoldMine to handle multiple contact sets.

If you will be developing an application to read and write to the GoldMine databases, we recommend that you use Dynamic Data Exchange (DDE) as described in Working with Dynamic Data Exchange (DDE) or the functions contained within GMXS32.DLL, as described in Using GMXS32.DLL for Database Access and Sync Log Updates. If you choose to write directly to our files without using DDE, you must be aware of the field/index structure and synchronization methodology used by GoldMine to ensure full compatibility.

To view how GoldMine uses RECTYPEs for various purposes, create a contact set, create sample contacts, and then create sample activities, and so on. Place obvious values in each of the fields. Use a database viewing utility, such as BR4, MS-Access, or Excel to view the sample records.

> **TIP:** Do not view your live contact database with an external application. Do not edit GoldMine fields with an external application.

CAL.DBF

Directory:	GMBASE
Description:	Calendar file—contains a record for each scheduled activity. The different record types are distinguished by the contents of the RECTYPE field. Different RECTYPEs may use each field for a different purpose.
Index File:	CAL.MDX

CAL Indexes

Name	Key
Cal	Rectype+userID+DTOS(onDate)+onTime
Calcont	AccountNo+rectype+DTOS(onDate)+onTime
Caldate	UserID+DTOS(onDate)+onTime
Calprob	Rectype+userID+Str(999-duration,3)
Calalarm	AlarmFlag+userID+DTOS(ALARMDATE)+alarmTime
Calrlink	lopRecID+RECTYPE+DTOS(ONDATE)+ONTIME
Calrecid	recId

CAL Structure

Field Name	Type	Len	Description
USERID	String	8	User Name
ACCOUNTNO	String	20	Account Number of linked contact
ONDATE	Date	8	Activity date
ONTIME	String	5	Activity Time
ENDDATE	Date	8	Ending Date of Scheduled Activity
ALARMFLAG	String	1	Alarm Flag
ALARMTIME	String	5	Alarm Time
ALARMDATE	Date	8	Alarm Date
ACTVCODE	String	3	Activity Code
RSVP	String	1	RSVP Notification

DURATION	Integer	3	Duration/Probability
RECTYPE	String	1	See: Rectype
ACONFIRM	String	3	Meeting Confirmation
APPTUSER	String	10	Meeting Confirmation User
STATUS	String	4	First character is flag, second char =1 if notes exist
DIRCODE	String	10	DirCode of the current contact file
NUMBER1	Integer	11	Sales Potential
NUMBER2	Integer	8	Units of a Forecasted Sale
COMPANY	String	60	Company/Contact Name
REF	String	80	Reference
NOTES	Memo	1	Notes
LINKRECID	String	15	Linked Record ID
ldoCrecid	String	15	Reserved for future use
LOPRECID	String	15	Linked Opportunity Manager Record ID
CREATEBY	String	8	Created by User
CREATEON	Date	8	Creation Date
CREATEAT	String	6	Creation Time
LASTUSER	String	8	Last Modified By
LASTDATE	Date	8	Last Modified Date
LASTTIME	String	5	Last Modified Time
RECID	String	15	Record ID

Rectype

The RECTYPE field contains the Calendar's activity type. The following values are possible contents of RECTYPE:

A	Appointment	F	Literature fulfillment	S	Sales potential
C	Call Back	M	Message	T	Next action
D	To-do	O	Other		
E	Event	Q	Queued e-mail		

CONTACT1.DBF

Directory:	COMMON
Description:	Contact file—contains the main fields of contact records
Index File:	CONTACT1.MDX

CONTACT1 Indexes

Name	Key
Contacc	AccountNo
Contcomp	Upper(company)+Substr(accountNo,10,4)
Contname	Upper(contact)+Substr(accountNo,10,4)
Contzip	zip+Substr(accountNo,10,4)
Contcity	Upper(city)+Substr(accountNo,10,4)
Contkey1	Upper(key1)+Substr(accountNo,10,4)
Contkey2	Upper(key2)+Substr(accountNo,10,4)
Contkey3	Upper(key3)+Substr(accountNo,10,4)
Contkey4	Upper(key4)+Substr(accountNo,10,4)
Contkey5	Upper(key5)+Substr(accountNo,10,4)
Contlast	Upper(lastName)+Substr(accountNo,10,4)
CONTSTAT	Upper(STATE+CITY)+SUBSTR(ACCOUNTNO,10,4)
CONTCNTY	UPPER(COUNTRY+STATE)+SUBTR(ACCOUNTNO,10,4)
Contphon	phone1+Substr(accountNo,10,4)
Cn1Recid	recid

CONTACT1 Relations

Related File->Field	Contact1 Field
Contact2->AccountNo	Contact1->AccountNo
ContHist->AccountNo	Contact1->AccountNo
ContSupp->AccountNo	Contact1->AccountNo
Cal->AccountNo	Contact1->AccountNo

CONTACT1 Structure

Field Name	Type	Len	Description
ACCOUNTNO	String	20	See: Account Number
COMPANY	String	40	Company Name
CONTACT	String	40	Contact Name
LASTNAME	String	15	Contact's Last Name
DEPARTMENT	String	35	Department
TITLE	String	35	Contact Title
SECR	String	20	Secretary
PHONE1	String	25	Phone 1
PHONE2	String	25	Phone 2
PHONE3	String	25	Phone 3
FAX	String	25	Fax
EXT1	String	6	Phone Extension 1
EXT2	String	6	Phone Extension 2
EXT3	String	6	FAX Extension used as EXT3 to maintain compatibility with previous versions
EXT4	String	6	Phone Extension 3
ADDRESS1	String	40	Address 1
ADDRESS2	String	40	Address 2
ADDRESS3	String	40	Address 3
CITY	String	30	City
STATE	String	20	State
ZIP	String	10	Zip Code
COUNTRY	String	20	Country
DEAR	String	20	Dear (Salutation)
SOURCE	String	20	Source (Lead)
KEY1	String	20	Key 1

KEY2	String	20	Key 2
KEY3	String	20	Key 3
KEY4	String	20	Key 4
KEY5	String	20	Key 5
STATUS	String	3	See : Internal Status
NOTES	Memo		Notes
MERGECODES	String	20	Merge Codes for primary contact
CREATEBY	String	8	Creation User
CREATEON	Date	8	Creation Date
CREATEAT	String	5	Creation Time
OWNER	String	8	Record Owner
LASTUSER	String	8	Last Modified By
LASTDATE	Date	8	Last Modified Date
LASTTIME	String	6	Last Modified Time
RECID	String	15	Record ID

Account Number

The ACCOUNTNO field contains the following information:

Positions	Value
1–6	Date in YYMMDD format
7–11	Seconds since midnight
12–17	Randomly generated
18–20	First three characters of the contact or company name

Internal Status

- *Position 1* of the Internal Status field keeps track of the type of phone number for the contact. If the first character is U, the phone numbers are formatted for USA-style phone numbers: (999)999-9999.
- *Position 2* indicates the curtain level (0=none, 1=partial, 2=full)
- *Position 3* indicates a record alert is present if the value is 1.

CONTACT2.DBF

Directory:	COMMON
Description:	Contact file—contains the additional fields of contact records. Each complete contact record has a record in this file. User-defined field data is stored in this file.
Index File:	CONTACT2.MDX

CONTACT2 Index

Name	Key
Contact2	accountNo
Cn2Recid	recId

CONTACT2 Structure

Field Name	Type	Len	Description
ACCOUNTNO	String	20	Account Number
CALLBACKON	Date	8	Call Back Date
CALLBACKAT	String	8	Call Back Time (unused compatibility field)
CALLBKFREQ	Smallint	3	Call Back Frequency
LASTCONTON	Date	8	Last Contact Date
LASTCONTAT	String	8	Last Contact Time
LASTATMPON	Date	8	Last Attempt Date
LASTATMPAT	String	8	Last Attempt Time
MEETDATEON	Date	8	Meeting Date
MEETTIMEAT	String	8	Meeting Time
COMMENTS	Date	65	Comments
PREVRESULT	String	65	Previous Results
NEXTACTION	String	65	Next Action
ACTIONON	Date	8	Next Action Date
CLOSEDATE	Date	8	Expected Close Date
USERDEF01	String	10	User Defined 1

USERDEF02	String	10	User Defined 2
USERDEF03	String	10	User Defined 3
USERDEF04	String	10	User Defined 4
USERDEF05	String	10	User Defined 5
USERDEF06	String	10	User Defined 6
USERDEF07	String	10	User Defined 7
USERDEF08	String	10	User Defined 8
USERDEF09	String	10	User Defined 9
USERDEF10	String	10	User Defined 10
RECID	String	15	Record ID

CONTGRPS.DBF

Directory:	COMMON
Description:	Groups file—the CONTGRPS file is used for both the group header, which defines each group, and members for each group.
Index File:	CONTGRPS.MDX

CONTGRPS Indexes

Name	Key
GroupNo	UPPER(userID+code)
GroupAcc	accountno+userID
GrpRecID	recId

CONTGRPS Structure (header records)

Field Name	Type	Len	Description
USERID	String	15	Group user
CODE	String	8	Group code
ACCOUNTNO	String	20	See: Header Info
REF	String	24	Group reference
RECID	String	15	Record ID/Group number

Header Info

The ACCOUNTNO field contains the following information when the CONTGRPS record is a group header record:

Positions	Value
1–8	"*M"
15–20	Total members in group

The next available group number is stored in the CODE field in the first physical record in CONTGRPS.DBF.

CONTGRPS Structure (member records)

Field Name	Type	Len	Description
USERID	String	15	Group number (from group header)
CODE	String	8	Member sort value
ACCOUNTNO	String	20	Linked contact accountno
REF	String	24	Member reference
RECID	String	15	Record ID

CONTHIST.DBF

Directory:	COMMON
Description:	Contact history file—contains a record for each completed activity
Index File:	CONTHIST.MDX

CONTHIST Indexes

Name	Key
ContHist	accountNo+DTOS(onDate)+RECID
ContHusr	USERID+SRECTYPE+DTOS(ONDATE)+RECID
CNHRLink	lopRecId+DTOS(ONDATE)
CnHRecid	recId

CONTHIST Structure

Field Name	Type	Len	Description
USERID	String	8	User
ACCOUNTNO	String	20	Account No.
SRECTYPE	String	1	First character of RecType
RECTYPE	String	10	See: Record Type
ONDATE	Date	8	Action Date
ONTIME	String	5	Action Time
ACTVCODE	String	3	Activity Code
RESULTCODE	String	3	Result Code
STATUS	String	2	First character is flag, second character =1 if notes exist
DURATION	String	8	Duration
UNITS	String	8	Units of a Forecasted Sale
REF	String	80	Reference
NOTES	Memo	1	Notes
LINKRECID	String	15	Linked Record ID
LOPRECID	String	15	Linked Opp. Mgr. Record
CREATEBY	String	8	Creation User
CREATEON	Date	8	Creation Date
CREATEAT	String	6	Creation Time
LASTUSER	String	8	Last Modified By
LASTDATE	Date	8	Last Modified Date
LASTTIME	String	6	Last Modified Time
RECID	String	15	Record ID
NUMBER1	Float		
NUMBER2	Float		

Record Type

The RECTYPE field contains the completed activity's type. The following values are possible contents of RECTYPE:

A	Appointment	M	Sent message	CI	Incoming call
C	Phone call	O	Other	CM	Returned message
D	To-do	S	Sale	CO	Outgoing call
E	Event	T	Next action	MG	E-mail message
F	Literature fulfillment	U	Unknown	MI	Received e-mail
L	Form	CC	Call back	MO	Sent e-mail

CONTSUPP.DBF

Directory:	COMMON
Description:	Supplementary contact set—contains a record for each additional contact record, referral and profile record. The different record types are distinguished by the contents of the RECTYPE field. Different RECTYPEs may use each field for a different purpose.
Index File:	CONTSUPP.MDX

CONTSUPP Indexes

Name	Key
ContSupp	accountNo+recType+UPPER(contact)
Contspfd	UPPER(RECTYPE+CONTACT+CONTSUPREF)
Cnsrecid	recId

CONTSUP Structure

Field Name	Type	Len	Description
ACCOUNTNO	String	20	Account No.
RECTYPE	String	1	See: Record Type
CONTACT	String	30	Contact Name/Profile
TITLE	String	35	Contact Title/Referral's Account Number

CONTSUPREF	String	35	Reference
DEAR	String	20	Dear (Salutation)
PHONE	String	20	Phone
EXT	String	6	Phone Extension
FAX	String	20	FAX number
LINKACCT	String	20	Linked Account
NOTES	Memo	1	Notes
ADDRESS1	String	40	Additional Contact Address 1
ADDRESS2	String	40	Additional Contact Address 2
ADDRESS3	String	40	Additional Contact Address 3
CITY	String	30	Additional Contact City
STATE	String	20	Additional Contact State
ZIP	String	10	Additional Contact Zip
COUNTRY	String	20	Additional Contact Country
MERGECODES	String	20	Merge Codes
STATUS	String	4	First character is flag, second char =1 if notes exist
LINKEDDOC	Memo	10	Linked Document
LASTUSER	String	8	Last Modified By
LASTDATE	Date	8	Last Modified Date
LASTTIME	String	5	Last Modified Time
RECID	String	15	Record ID

Record Type

The **RECTYPE** field contains the record type. The following values are possible contents of **RECTYPE:**

C	Additional contact record	O	Organizational chart
E	Automated Process attached event	P	Profile record/extended profile record
H	Extended profile header	R	Referral record
L	Linked document		

The *RECTYPE* value *H* can be linked to records with the RECTYPE value *P*. Assigning extended information settings to a profile (assigned to a tab, or extended fields used) creates an *H* record type to store the settings. The profile record stores a character string in the *Phone* field that matches the *H* record's *ACCOUNTNO* field

INFOMINE.DBF

Directory:	GMBASE
Description:	InfoCenter file—stores all data for the InfoCenter
Index File:	INFOMINE.MDX

INFOMINE Indexes

Name	Key
infomine	UPPER(rectype+LEFT(TSECTION,80)+LEFT(TOPIC,10)
infosort	sortKey
infotran	recType+recID
infrecid	recId

INFOMINE Structure

Field Name	Type	Len	Description
ACCOUNTNO	String	20	Account No.
CREATEBY	String	8	Creation User
RECTYPE	String	10	Record Type
SORTKEY	String	20	Sort Key
TSECTION	String	100	Section
TOPIC	String	80	Topic
KEYWORDS	String	80	Keywords
OPTIONS	String	10	Options
OPTIONS1	String	20	Options1
OPTIONS2	String	20	Options2
LINKEDDOC	Memo	1	Linked Document
NOTES	Memo	1	Notes

USERREAD	String	8	Read Access
USERWRITE	String	8	Write Access
LASTUSER	String	8	Last Modified By
LASTDATE	Date	8	Last Modified Date
LASTTIME	String	5	Last Modified Time
RECID	String	15	Record ID

LOOKUP.DBF

Directory:	GMBASE
Description:	Lookup file—contains a record of each defined look-up entry
Index File:	LOOKUP.MDX

LOOKUP Indexes

Name	***Key***
Lookup	UPPER(FIELDName+entry)
lkurecid	recId

LOOKUP Structure

Field Name	**Type**	**Len**	**Description**
FIELDNAME	String	11	Field Name
LOOKUPSUPP	String	10	Lookup Options
ENTRY	String	40	Description
RECID	String	15	Record ID

MAILBOX.DBF

Directory:	GMBASE
Description:	E-mail Center mailbox file—stores all GoldMine e-mail
Index File:	MAILBOX.MDX

MAILBOX Indexes

Name	Key
mboxlink	LinkRecId
mboxuser	userId+folder+FOLDER2+DTOS(MAILDATE)
mbxrecid	recId

MAILBOX Structure

Field Name	Type	Len	Description
LINKRECID	String	15	Linked Record ID
FLAGS	String	8	See: Flags
USERID	String	8	User Name
FOLDER	String	20	See: Folder
FOLDER2	String	20	Subfolder
ACCOUNTNO	String	20	Account No.
CREATEON	Date	8	Creation Date
MAILSIZE	String	8	Mail Size
MAILDATE	Date		Mail Date
MAILTIME	String	8	Mail Time
MAILREF	String	100	Reference
RFC822	Memo	1	Entire Mail Message
RECID	String	15	Record ID

Flags

The FLAGS field is a String type, but actually stores a number. When the number is converted to binary, the following rules apply:

Bit	On	Off
1	Read	Not Read
2	In History	Not in History

3	Outbound	Inbound
4	Attachments	No Attachments

Folder

The **FOLDER** field contains the name of the folder in which mail is stored. GoldMine uses the following predefined folders:

X-GM-INBOX	-Inbox
X-GM-OUTBOX	-Outbox
X-GM-TEMPLATES	-Templates

OPMGR.DBF

Directory:	GMBASE
Description:	Opportunity Manager file—stores all data maintained in the Opportunity Manager
Index File:	OPMGR.MDX

OPMGR Indexes

Name	Key
OpMgr	UPPER(recType+userID+stage)
OpId	opId+recType
OPACCNO	ACCOUNTNO+RECTYPE+OPID
OpRecID	recID

OPMGR Structure

Field Name	Type	Len	Description
OPID	String	15	Opportunity ID
RECTYPE	String	3	
ACCOUNTNO	String	20	Account No.
USERID	String	8	User Name
FLAGS	String	10	Flags
COMPANY	String	40	Company

CONTACT	String	40	Contact
NAME	String	50	Name
STATUS	String	50	Status
CYCLE	String	50	Cycle
STAGE	String	30	Stage
SOURCE	String	30	Source
F1	String	20	
F2	String	20	
F3	String	10	
STARTDATE	Date	8	Start Date
CLOSEDDATE	Date	8	Close Date
CLOSEBY	Date	8	Close by
FORAMT	Float	10	For Amount
FORPROB	Integer	4	Probability
CLOSEAMT	Float	10	Close Amount
Notes	Memo	1	Notes
RECID	String	15	Record ID

Record Type

The following OpMgr rectypes are valid, where x represents O for opportunity records, or P for project records:

O	Opportunity header record	xT	Team member
P	Project header record	xI	Issue
*x*C	Contact	xF	Field
*x*P	Competitor	xK	Task

PERPHONE.DBF

Directory:	GMBASE
Description:	Personal Rolodex file—contains a record of each entry in the user's Rolodex

Index File:	PERPHONE.MDX

PERPHONE Indexes

Name	Key
Perphone	UPPER(recType+userID+contact)
pphrecid	recId

PERPHONE Structure

Field Name	Type	Len	Description
RECTYPE	String	1	Record Type
USERID	String	8	User Name
STATUS	String	2	Status
CONTACT	String	30	Contact Name
PHONE1	String	16	Phone Number
RECID	String	15	Record ID

RESITEMS.DBF

Directory:	GMBASE
Description:	Resources file—stores data regarding equipment, facilities, and other resources that you can schedule from the Resources' Master File.
Index File:	RESITEMS.MDX

RESITEMS Indexes

Name	Key
resource	name
rscrecid	recid

RESITEMS Structure

Field Name	Type	Len	Description
NAME	String	8	Name

CODE	String	10	Code
RESDESC	String	40	Description
CUSTODIAN	String	8	Custodian
NOTES	Memo	1	Notes
RECID	String	15	Record ID

SPFILES.DBF

Directory:	GMBASE
Description:	Contact files directory—contains a record for each GoldMine contact set
Index File:	SPFILES.MDX

SPFILES Index

Name	Key
Spfiles	UPPER(dirPath)
Sflcode	dirCode
sflrecid	recId

SPFILES Structure

Field Name	Type	Len	Description
DIRNAME	String	35	Contact file description
DIRPATH	String	100	Contact file path
USERID	String	8	Contact file user
DIRCODE	String	10	Contact Set Code
DBPASSWORD	String	36	Database Password
DRIVER	String	25	Database Driver
RECID	String	15	Record ID

SQL Database Structures

Overview

Third-party developers are encouraged to integrate their products with GoldMine, thereby enhancing both products. If you design a commercial program that works with GoldMine, please contact GoldMine so we can include your program in our Enhancement Guide.

This chapter describes the file organization and structures of Goldmine SQL format databases in an SQL format. Each database file is listed separately and includes its associated index files, database structure, and special notes. For information about working with the GoldMine Xbase format database, see Xbase Database Structures. The following pages describe the database structures of most GoldMine .DBF files. This chapter does not include a discussion of every database. Security and system database files are not included in this section. You should not interface with these files. For an in-depth discussion on interfacing with GoldMine, visit the Public.GoldMine.Programming newsgroup, which you can access directly from http://www.goldmine.com/.

If you will be developing an application to read and write to the GoldMine databases, we recommend that you use Dynamic Data Exchange (DDE) as described in Working with Dynamic Data Exchange (DDE) or the functions contained within `GMXS32.DLL`, as described in Using GMXS32.DLL for Database Access and Sync Log Updates. If you choose to write directly to our files without using DDE, you must be aware of the field/index structure and synchronization methodology used by GoldMine to ensure full compatibility.

To view how GoldMine uses RECTYPEs for various purposes, create a contact set, create sample contacts, and then create sample activities, and so on. Place obvious values in each of the fields. Use a database viewing utility, such as MS-Access, MSSQL Enterprise Manager, or isql to view the sample records.

> **TIP:** Do not view your live contact database with an external application. Do not edit GoldMine fields with an external application.

CAL Table

Description:	Calendar file—contains a record for each scheduled activity. The different record types are distinguished by the contents of the RECTYPE field. Different RECTYPEs may use each field for a different purpose.

CAL Indexes

Name	Index Tags	Unique?
CALCONT	ACCOUNTNO+RECTYPE+ONDATE+ONTIME+RECID	No
CAL	RECTYPE+USERID+ONDATE+ONTIME+RECID	No
CALDATE	USERID+ONDATE+ONTIME+RECID	No
CALPROB	RECTYPE+USERID	No
CALALARM	ALARMFLAG+USERID+ALARMDATE+ALARMTIME	No
CALRLINK	LOPRECID+RECTYPE+ONDATE+ONTIME	No
CALRECID	RECID	Yes

CAL Structure

Field Name	Type	Len	Description
USERID	String	8	User Name
ACCOUNTNO	String	20	Account Number of linked contact
ONDATE	Date	8	Activity date
ONTIME	String	5	Activity Time
ENDDATE	Date	8	Ending Date of Scheduled Activity
ALARMFLAG	String	1	Alarm Flag
ALARMTIME	String	5	Alarm Time
ALARMDATE	Date	8	Alarm Date
ACTVCODE	String	3	Activity Code
RSVP	String	1	RSVP Notification
DURATION	Integer	3	Duration/Probability
RECTYPE	String	1	See: Record Type
ACONFIRM	String	3	Meeting Confirmation
APPTUSER	String	10	Meeting Confirmation User
STATUS	String	4	First character is flag, second char =1 if notes exist
DIRCODE	String	10	DirCode of the current contact file

NUMBER1	Integer	11	Sales Potential
NUMBER2	Integer	8	Units of a Forecasted Sale
COMPANY	String	60	Company/Contact Name
REF	String	80	Reference
NOTES	Memo	1	Notes
LINKRECID	String	15	Linked Record ID
ldoCrecid	String	15	Reserved for future use
LOPRECID	String	15	Linked Opportunity Manager Record ID
CREATEBY	String	8	Created by User
CREATEON	Date	8	Creation Date
CREATEAT	String	6	Creation Time
LASTUSER	String	8	Last Modified By
LASTDATE	Date	8	Last Modified Date
LASTTIME	String	5	Last Modified Time
RECID	String	15	Record ID

Record Type

The *RECTYPE* field contains the calendar's activity type. The following values are possible contents of *RECTYPE*:

A	Appointment	***F***	Literature fulfillment	***S***	Sales potential
C	Call Back	***M***	Message	***T***	Next action
D	To-do	***O***	Other		
E	Event	***Q***	Queued e-mail		

CONTACT1 Table

Description:	Contact file—contains the main fields of contact records

CONTACT1 Indexes

Name	Index Tags	Unique?

CONTACC	ACCOUNTNO	No
CONTCNTY	U_COUNTRY+U_STATE+ACCOUNTNO	No
CONTCOMP	U_COMPANY+ACCOUNTNO	No
CONTNAME	U_CONTACT+ACCOUNTNO	No
CONTZIP	ZIP+ACCOUNTNO	No
CONTCITY	U_CITY+ACCOUNTNO	No
CONTKEY1	U_KEY1+ACCOUNTNO	No
CONTKEY2	U_KEY2+ACCOUNTNO	No
CONTKEY3	U_KEY3+ACCOUNTNO	No
CONTKEY4	U_KEY4+ACCOUNTNO	No
CONTKEY5	U_KEY5+ACCOUNTNO	No
CONTLAST	U_LASTNAME+ACCOUNTNO	No
CONTSTAT	U_STATE+U_CITY+ACCOUNTNO	No
CONTPHON	PHONE1+ACCOUNTNO	No
CN1RECID	RECID	Yes

CONTACT1 Relations

Related File->Field	Contact1 Field
Contact2->AccountNo	Contact1->AccountNo
ContHist->AccountNo	Contact1->AccountNo
ContSupp->AccountNo	Contact1->AccountNo
Cal->AccountNo	Contact1->AccountNo

CONTACT1 Structure

Field Name	Type	Len	Description
ACCOUNTNO	String		See: Account Number
COMPANY	String	40	Company Name
CONTACT	String	40	Contact Name
LASTNAME	String	15	Contact's Last Name

DEPARTMENT	String	35	Department
TITLE	String	35	Contact Title
SECR	String	20	Secretary
PHONE1	String	25	Phone 1
PHONE2	String	25	Phone 2
PHONE3	String	25	Phone 3
FAX	String	25	Fax
EXT1	String	6	Phone Extension 1
EXT2	String	6	Phone Extension 2
EXT3	String	6	FAX Extension used as EXT3 to maintain compatibility with previous versions
EXT4	String	6	Phone Extension 3
ADDRESS1	String	40	Address 1
ADDRESS2	String	40	Address 2
ADDRESS3	String	40	Address 3
CITY	String	30	City
STATE	String	20	State
ZIP	String	10	Zip Code
COUNTRY	String	20	Country
DEAR	String	20	Dear (Salutation)
SOURCE	String	20	Source (Lead)
KEY1	String	20	Key 1
KEY2	String	20	Key 2
KEY3	String	20	Key 3
KEY4	String	20	Key 4
KEY5	String	20	Key 5
STATUS	String	3	See: Internal Status
NOTES	Memo		Notes

MERGECODES	String	20	Merge Codes for primary contact
CREATEBY	String	8	Creation User
CREATEON	Date	8	Creation Date
CREATEAT	String	5	Creation Time
OWNER	String	8	Record Owner
LASTUSER	String	8	Last Modified By
LASTDATE	Date	8	Last Modified Date
LASTTIME	String	6	Last Modified Time
U_COMPANY	String	40	Upper-case shadow of Company field
U_CONTACT	String	40	Upper-case shadow of Contact field
U_LASTNAME	String	15	Upper-case shadow of contact's Last Name field
U_CITY	String	30	Upper-case shadow of City field
U_STATE	String	20	Upper-case shadow of State field
U_COUNTRY	String	20	Upper-case shadow of Country field
U_KEY1	String	20	Upper-case shadow of Key 1 field
U_KEY2	String	20	Upper-case shadow of Key 2 field
U_KEY3	String	20	Upper-case shadow of Key 3 field
U_KEY4	String	20	Upper-case shadow of Key 4 field
U_KEY5	String	20	Upper-case shadow of Key 5 field
RECID	String	15	Record ID

Account Number

The ACCOUNTNO field contains the following information:

Positions	Value
1–6	Date in YYMMDD format
7–11	Seconds since midnight
12–17	Randomly generated
18–20	First three characters of the contact or company name

Internal Status

- *Position 1* of the Internal Status field keeps track of the type of phone number for the contact. If the first character is U, the phone numbers are formatted for USA-style phone numbers: (999)999-9999.
- *Position 2* indicates the curtain level (0=none, 1=partial, 2=full).
- *Position 3* indicates a record alert is present if the value is 1.

CONTACT2 Table

Description:	Contact file—contains the additional fields of contact records. Each complete contact record has a record in this file. User-defined field data is stored in this file.

CONTACT2 Index

Name	Index Tags	Unique?
CONTACT2	ACCOUNTNO	No
CN2RECID	RECID	Yes

CONTACT2 Structure

Field Name	Type	Len	Description
ACCOUNTNO	String	20	Account Number
CALLBACKON	Date	8	Call Back Date
CALLBACKAT	String	8	Call Back Time (unused compatibility field)
CALLBKFREQ	Smallint	3	Call Back Frequency
LASTCONTON	Date	8	Last Contact Date
LASTCONTAT	String	8	Last Contact Time
LASTATMPON	Date	8	Last Attempt Date
LASTATMPAT	String	8	Last Attempt Time
MEETDATEON	Date	8	Meeting Date
MEETTIMEAT	String	8	Meeting Time
COMMENTS	Date	65	Comments
PREVRESULT	String	65	Previous Results
NEXTACTION	String	65	Next Action
ACTIONON	Date	8	Next Action Date

CLOSEDATE	Date	8	Expected Close Date
USERDEF01	String	10	User Defined 1
USERDEF02	String	10	User Defined 2
USERDEF03	String	10	User Defined 3
USERDEF04	String	10	User Defined 4
USERDEF05	String	10	User Defined 5
USERDEF06	String	10	User Defined 6
USERDEF07	String	10	User Defined 7
USERDEF08	String	10	User Defined 8
USERDEF09	String	10	User Defined 9
USERDEF10	String	10	User Defined 10
RECID	String	15	Record ID

CONTGRPS Table

Description:	Groups file—the CONTGRPS file is used for both the group header, which defines each group, and members for each group.

CONTGRPS Indexes

Name	Index Tags	Unique?
GROUPNO	USERID+U_CODE+RECID	No
GROUPACC	ACCOUNTNO+USERID	No
GRPRECID	RECID	Yes

CONTGRPS Structure (header records)

Field Name	Type	Len	Description
USERID	String	15	Group user
CODE	String	8	Group code
ACCOUNTNO	String	20	See: Header Info
REF	String	24	Group reference

U_CODE	String	8	Upper-case shadow of member sort value
RECID	String	15	Record ID/Group number

Header Info

The **ACCOUNTNO** field contains the following information when the CONTGRPS record is a group header record:

Positions	Value
1–8	"*M"
15–20	Total members in group

The next available group number is stored in the *CODE* field in the first physical record in CONTGRPS.DBF.

CONTGRPS Structure (member records)

Field Name	Type	Len	Description
USERID	String	15	Group number (from group header)
CODE	String	8	Member sort value
ACCOUNTNO	String	20	Linked contact accountno
REF	String	24	Member reference
U_CODE	String	8	Upper-case shadow of member sort value
RECID	String	15	Record ID

CONTHIST Table

Description:	Contact history file—contains a record for each completed activity

CONTHIST Indexes

Name	Index Tags	Unique?
CONTHIST	ACCOUNTNO+ONDATE+RECID	No
CONTHUSR	USERID+SRECTYPE+ONDATE+RECID	No
CNHRLINK	LOPRECID+ONDATE	No
CNHRECID	RECID	Yes

CONTHIST Structure

Field Name	Type	Len	Description
USERID	String	8	User
ACCOUNTNO	String	20	Account No.
SRECTYPE	String	1	First character of RecType
RECTYPE	String	10	See: Record Type
ONDATE	Date	8	Action Date
ONTIME	String	5	Action Time
ACTVCODE	String	3	Activity Code
RESULTCODE	String	3	Result Code
STATUS	String	2	First character is flag, second character =1 if notes exist
DURATION	String	8	Duration
UNITS	String	8	Units of a Forecasted Sale
REF	String	80	Reference
Field Name	Type	Len	Description
NOTES	Memo	1	Notes
LINKRECID	String	15	Linked Record ID
LOPRECID	String	15	Linked Opp. Mgr. Record
CREATEBY	String	8	Creation User
CREATEON	Date	8	Creation Date
CREATEAT	String	6	Creation Time
LASTUSER	String	8	Last Modified By
LASTDATE	Date	8	Last Modified Date
LASTTIME	String	6	Last Modified Time
RECID	String	15	Record ID
NUMBER1	Float		
NUMBER2	Float		

Record Type

The **RECTYPE** field contains the record type. The following values are possible contents of *RECTYPE*:

C	Additional contact record	***O***	Organizational chart
E	Automated Process attached event	***P***	Profile record/extended profile record
H	Extended profile header	***R***	Referral record
L	Linked document		

The *RECTYPE* value *H* can be linked to records with the RECTYPE value *P*. Assigning extended information settings to a profile (assigned to a tab or extended fields used) creates an *H* record type to store the settings. The profile record stores a character string in the *PHONE* field that matches the *H* record's *ACCOUNTNO* field.

CONTSUPP Table

Description: Supplementary contact set—contains a record for each additional contact record, referral and profile record. The different record types are distinguished by the contents of the RECTYPE field. Different RECTYPEs may use each field for a different purpose.

CONTSUPP Indexes

Name	Index Tags	Unique?
CONTSUPP	ACCOUNTNO+RECTYPE+U_ CONTACT+RECID	No
CONTSPFD	RECTYPE+U_CONTACT+U_CONTSUPREF	No
CNSRECID	RECID	Yes

CONTSUPP Structure

Field Name	Type	Len	Description
ACCOUNTNO	String	20	Account No.
RECTYPE	String	1	See: Record Type
CONTACT	String	30	Contact Name/Profile
TITLE	String	35	Contact Title/Referral's Account Number
CONTSUPREF	String	35	Reference
DEAR	String	20	Dear (Salutation)

PHONE	String	20	Phone
EXT	String	6	Phone Extension
FAX	String	20	FAX number
LINKACCT	String	20	Linked Account
NOTES		1	Notes
ADDRESS1	String	40	Additional Contact Address 1
ADDRESS2	String	40	Additional Contact Address 2
ADDRESS3	String	40	Additional Contact Address 3
CITY	String	30	Additional Contact City
STATE	String	20	Additional Contact State
ZIP	String	10	Additional Contact Zip
COUNTRY	String	20	Additional Contact Country
MERGECODES	String	20	Merge Codes
STATUS	String	4	First character is flag, second char =1 if notes exist
LINKEDDOC	Memo	10	Linked Document
LASTUSER	String	8	Last Modified By
LASTDATE	Date	8	Last Modified Date
LASTTIME	String	5	Last Modified Time
U_CONTACT	String	30	Upper-case shadow of Contact field
U_CONTSUPREF	String	35	Upper-case shadow of Reference field
RECID	String	15	Record ID

Record Type

The **RECTYPE** field contains the record type. The following values are possible contents of *RECTYPE*:

C	Additional contact record	***O***	Organizational chart
E	Automated Process attached event	***P***	Profile record/extended profile record
H	Extended profile header	***R***	Referral record
L	Linked document		

The *RECTYPE* value *H* can be linked to records with the RECTYPE value *P*. Assigning extended information settings to a profile (assigned to a tab or extended fields used) creates an *H* record type to store the settings. The profile record stores a character string in the *PHONE* field that matches the *H* record's *ACCOUNTNO* field.

INFOMINE Table

Description:	InfoCenter file—stores all data for the InfoCenter

INFOMINE Indexes

Name	Index Tags	Unique?
INFOMINE	RECTYPE+U_TSECTION+U_TOPIC	No
INFOSORT	SORTKEY	No
INFOTRAN	RECTYPE+RECID	No
INFRECID	RECID	Yes

INFOMINE Structure

Field Name	Type	Len	Description
ACCOUNTNO	String	20	Account No.
CREATEBY	String	8	Creation User
RECTYPE	String	10	Record Type
SORTKEY	String	20	Sort Key
TSECTION	String	100	Section
TOPIC	String	80	Topic
KEYWORDS	String	80	Keywords
OPTIONS	String	10	Options
OPTIONS1	String	20	Options1
OPTIONS2	String	20	Options2
LINKEDDOC	Memo	1	Linked Document
NOTES	Memo	1	Notes
USERREAD	String	8	Read Access
USERWRITE	String	8	Write Access

LASTUSER	String	8	Last Modified By
LASTDATE	Date	8	Last Modified Date
LASTTIME	String	5	Last Modified Time
U_TSECTION	String	100	Upper-case shadow of Section field
U_TOPIC	String	80	Upper-case shadow of Topic field
RECID	String	15	Record ID

LOOKUP Table

Description: Lookup file—contains a record of each defined look-up entry

LOOKUP Indexes

Name	Index Tags	Unique?
LOOKUP	FIELDNAME+U_ENTRY	No
LKURECID	RECID	Yes

LOOKUP Structure

Field Name	Type	Len	Description
FIELDNAME	String	11	Field Name
LOOKUPSUPP	String	10	Lookup Options
ENTRY	String	40	Description
U_ENTRY	String	40	Upper-case shadow of Description field
RECID	String	15	Record ID

MAILBOX Table

Description: E-mail Center mailbox file—stores all GoldMine e-mail

MAILBOX Indexes

Name	Index Tags	Unique?
MBOXLINK	LINKRECID	No
MBOXUSER	USERID+FOLDER+FOLDER2+MAILDATE	No

MBXRECID	RECID	Yes

MAILBOX Structure

Field Name	Type	Len	Description
LINKRECID	String	15	Linked Record ID
FLAGS	String	8	See: Flags
USERID	String	8	User Name
FOLDER	String		See: Folder
FOLDER2	String	20	Subfolder
ACCOUNTNO	String	20	Account No.
CREATEON	Date	8	Creation Date
MAILSIZE	String	8	Mail Size
MAILDATE	Date		Mail Date
MAILTIME	String	8	Mail Time
MAILREF	String	100	Reference
RFC822	Memo	1	Entire Mail Message
RECID	String	15	Record ID

Flags

The *FLAGS* field is a String type, but actually stores a number. When the number is converted to binary, the following rules apply:

Bit	On	Off
1	Read	Not Read
2	In History	Not in History
3	Outbound	Inbound
4	Attachments	No Attachments

Folder

The *FOLDER* field contains the name of the folder in which mail is stored. GoldMine uses the following predefined folders:

X-GM-INBOX	-Inbox
X-GM-OUTBOX	-Outbox
X-GM-TEMPLATES	-Templates

OPMGR Table

Description: Opportunity Manager file—stores all data maintained in the Opportunity Manager

OPMGR Indexes

Name	Index Tags	Unique?
OPMGR	RECTYPE+USERID+U_STAGE	No
OPID	OPID+RECTYPE	No
OPACCNO	ACCOUNTNO+RECTYPE+OPID	No
OPRECID	RECID	Yes

OPMGR Structure

Field Name	Type	Len	Description
OPID	String	15	Opportunity ID
RECTYPE	String	3	
ACCOUNTNO	String	20	Account No.
USERID	String	8	User Name
FLAGS	String	10	Flags
COMPANY	String	40	Company
CONTACT	String	40	Contact
NAME	String	50	Name
STATUS	String	50	Status
CYCLE	String	50	Cycle
STAGE	String	30	Stage
SOURCE	String	30	Source

F1	String	20	
F2	String	20	
F3	String	10	
STARTDATE	Date	8	Start Date
CLOSEDDATE	Date	8	Close Date
CLOSEBY	Date	8	Close by
FORAMT	Float	10	For Amount
FORPROB	Integer	4	Probability
CLOSEAMT	Float	10	Close Amount
Notes	Memo	1	Notes
U_STAGE	String	30	Upper-case shadow of Stage field
RECID	String	15	Record ID

Record Type

The following OpMgr rectypes are valid, where x represents O for opportunity records, or P for project records:

O	Opportunity header record	***xT***	Team member
P	Project header record	***xI***	Issue
xC	Contact	***xF***	Field
xP	Competitor	***xK***	Task

PERPHONE Table

Description:	Personal Rolodex file—contains a record of each entry in the user's Rolodex

PERPHONE Indexes

Name	Index Tags	Unique?
PERPHONE	RECTYPE+USERID+U_CONTACT	No
PPHRECID	RECID	Yes

PERPHONE Structure

Field Name	Type	Len	Description
RECTYPE	String	1	Record Type
USERID	String	8	User Name
STATUS	String	2	Status
CONTACT	String	30	Contact Name
PHONE1	String	16	Phone Number
U_CONTACT	String	30	Upper-case shadow of Contact field
RECID	String	15	Record ID

RESITEMS Table

Description:	Resources file—stores data regarding equipment, facilities, and other resources that you can schedule from the Resources' Master File.

RESITEMS Indexes

Name	Index Tags	Unique?
RESITEMS	NAME	No
RSRECID	RECID	Yes

RESITEMS Structure

Field Name	Type	Len	Description
NAME	String	8	Name
CODE	String	10	Code
RESDESC	String	40	Description
CUSTODIAN	String	8	Custodian
NOTES	Memo	1	Notes
RECID	String	15	Record ID

SPFILES Table

Description:	Contact files directory—contains a record for each GoldMine contact set

SPFILES Index

Name	Index Tags	Unique?
SFLCODE	DIRCODE	No
SFLRECID	RECID	Yes
SPFILES	U_DIRPATH	No

SPFILES Structure

C	Type	Len	Description
DIRNAME	String	35	Contact file description
DIRPATH	String	100	Contact file path
USERID	String	8	Contact file user
DIRCODE	String	10	Contact Set Code
DBPASSWORD	String	36	Database Password
DRIVER	String	25	Database Driver
U_DIRPATH	String	100	Upper-case shadow of Contact file path
RECID	String	15	Record ID

Appendix: Code Examples

Overview

This appendix contains code examples for the `GMXS32.DLL` and `GMXMLAPI.DLL` in the following programming languages:

- C++
- Visual Basic
- Delphi

GMXS32.DLL Code Examples

This section shows sample codes for C++, Visual Basic, and Delphi.

C++ Examples

The following C++ files have been provided as part of this package:

GM5S32.H: C Header file containing all of the `GMXS32.DLL` function prototypes.

Function prototypes

```
////////////////////////
//
// gm5s32.h
// Purpose : GM5S32.DLL interface
#ifndef __GM5S32_H
#define __GM5S32_H
#ifdef __cplusplus
 extern "C" {
#endif
// licensing structure passed to GMW_GetLicenseInfo
typedef struct
 {
 char szLicensee[60]; // licensee name
 char szLicNo[20]; // master serial number
 char szSiteName[20]; // undocked site name
 long iLicUsers; // licensed users
```

```
 long iSQLUsers; // licensed SQL users
 long iGSSites; // license GoldSync sites
 long isDemo; // is demo install
 long isServerLic; // is primary license ('D' or 'E')
 long isRemoteLic; // is remote license ('U' or 'S')
 long isUSALicense; // is USA license (1=US,128/32
 // bit, 0=nonUS, 32-bit only)
 long iDLLVersion; // the DLL version (400822)
long iReserved1;
 long iReserved2;
 long szReserved[100];
} GMW_LicInfo;
// DLL initialization functions
int _stdcall GMW_LoadBDE( char *szSysDir, char *szGoldDir, char
*szCommonDir, char *szUser =0, char *szPass =0 );
int _stdcall GMW_UnloadBDE();
int _stdcall GMW_SetSQLUserPass( char *szUserName, char *szPassword );
int _stdcall GMW_GetLicenseInfo( GMW_LicInfo *pLic );
long _stdcall GMW_IsUserGroupMember( char *szGroup, char *szUserID );
// DataStream functions
// DBF workarea functions
long _stdcall GMW_DB_Open( char *szTableName );
long _stdcall GMW_DB_Close( long pArea );
long _stdcall GMW_DB_Append( long pArea, char* szRecID );
long _stdcall GMW_DB_Replace( long pArea, char* szField, char *szData, int
iAddTo );
long _stdcall GMW_DB_Delete( long pArea );
long _stdcall GMW_DB_Unlock( long pArea );
long _stdcall GMW_DB_Read( long pArea, char *szField, char *szBuf, int
iBufSize );
long _stdcall GMW_DB_Top ( long pArea );
long _stdcall GMW_DB_Bottom( long pArea );
long _stdcall GMW_DB_SetOrder( long pArea, char *szTag );
long _stdcall GMW_DB_Seek( long pArea, char* szParam );
long _stdcall GMW_DB_Skip( long pArea, int nSkip =1 );
long _stdcall GMW_DB_Goto( long pArea, char *szRecNo );
long _stdcall GMW_DB_Move( long pArea, char *szCommand, char* szParam );
long _stdcall GMW_DB_Search( long pArea, char *szExpr, char *szRecID );
long _stdcall GMW_DB_Filter( long pArea, char *szFilterExpr );
long _stdcall GMW_DB_Range( long pArea, char *szMin, char* szMax, char*
szTag );
long _stdcall GMW_DB_RecNo( long pArea, char *szRecID );
long _stdcall GMW_DB_IsSQL( long pArea );
// Quick one-field access functions
// (these are slow -- do not use in loops)
long _stdcall GMW_DB_QuickSeek( char *szTableName, char *szIndex, char
*szSeekValue, char *szRecID );
long _stdcall GMW_DB_QuickRead( char *szTableName, char *szRecID, char
*szField, char *szValue, int iLen );
long _stdcall GMW_DB_QuickReplace( char *szTableName, char *szRecID, char
*szField, char *szValue, int iAddTo =0 );
```

```
// Sync functions
int _stdcall GMW_SyncStamp( char *szStamp, char *szOutBuf );
int _stdcall GMW_UpdateSyncLog( char *szTable, char *szRecID,
char *szField, char *szAction );
int _stdcall GMW_ReadImpTLog( char *szFile, int bDelWhenDone, char
*szStatus );
char* _stdcall GMW_NewRecID( char *pBuff, char *pUser );

// misc functions
long _stdcall GMW_UserAccess( long iOption );
struct GMWnv;
typedef GMWnv *HGMNV;
// GM5S32.DLL business logic functions
long _stdcall GMW_Execute( const char *szFunc, HGMNV hgmnv );
// create, release & copy name value containers
HGMNV __stdcall GMW_NV_Create();
HGMNV __stdcall GMW_NV_CreateCopy(HGMNV hgmnv);
void __stdcall GMW_NV_Delete(HGMNV hgmnv);
void __stdcall GMW_NV_Copy(HGMNV hgmnvDestination , HGMNV hgmnvSource);
// get and set value by name
const char* __stdcall GMW_NV_GetValue(HGMNV hgmnv, const char* name, const
char* defaultValue);
void __stdcall GMW_NV_SetValue(HGMNV hgmnv, const char* name, const char*
value);
// Check if name exists. returns: 0 failed, 1 success
long __stdcall GMW_NV_NameExists(HGMNV hgmnv, const char* name);
// remove name(s)
void __stdcall GMW_NV_EraseName(HGMNV hgmnv, const char* name);
void __stdcall GMW_NV_EraseAll(HGMNV hgmnv);
// iterate over name-value list (1 based)
long __stdcall GMW_NV_Count(HGMNV hgmnv);
const char* __stdcall GMW_NV_GetNameFromIndex(HGMNV hgmnv, long index);
const char* __stdcall GMW_NV_GetValueFromIndex(HGMNV hgmnv, long index);
void __stdcall GMW_NV_SetStr(HGMNV hgmnv, char dlmName, char dlmVal,const
char* pszStr);

#ifdef __cplusplus
 /* close extern "C" { */
 }
#endif
#endif // __GM5S32_H
```

Logging In

The following example uses C++ to access the GM5S32.DLL functions The DLL is dynamically loaded and its function addresses are retrieved using the GetProcAddress API.

```
// prototypes
typedef int (*fnGMW_LoadBDE) ( char *szSysDir, char *szGoldDir, char
*szCommonDir, char *szUser );
typedef int (*fnGMW_UnloadBDE) ();
void GM5S32_DLL_Test()
```

```
{
 // load the GM5S32.DLL
 HMODULE hLib = LoadLibrary("GM5S32.DLL");
 if( hLib )
 {
// get proc addresses of GM5S32 functions
 fnGMW_LoadBDE GMW_LoadBDE = (fnGMW_LoadBDE) GetProcAddress(
(HINSTANCE) hLib,"GMW_LoadBDE");
fnGMW_UnloadBDE GMW_UnloadBDE = (fnGMW_UnloadBDE)
GetProcAddress((HINSTANCE) hLib,"GMW_UnloadBDE");
// initialize the API
 GMW_LoadBDE( "d:\\gm4", "d:\\gm4", "d:\\gm4\\demo", szUser, szPass );
 // do whatever.............
 // shut down API
 GMW_UnloadBDE();
// unload the DLL
 FreeLibrary(hLib);
 }
return;
}
```

Creating a Contact with Business Logic/ Enumerating a Name Value Container/DataStream

The following DataStream example assumes that `GM5S32.DLL` has already been loaded and the function addresses have been retrieved. The first example retrieves a relatively small number of records into a fixed-size packet buffer, while the second example retries a large number of records using 100-record packet buffers.

```
void DataStreamDLL_Example()
{
long iHandle = 0;
long iOK = 0;
// Example 1:
// Get a small number of records and use a fixed size buffer
//
// return all contact names at GoldMine Inc.
//
char *szSQL1 = "SELECT Contact FROM Contact1 "
"WHERE U_COMPANY LIKE 'GOLDMINE INC.%' "
"ORDER BY U_CONTACT";
// send DataStream SQL Query
if( (iHandle = GMW_DS_Query( szSQL1 )) > 0 )
 {
 // allocate buffer for 200 contacts at 40 chars per/name
 long iBufSize = 200*40 +20;
char *szBuf = new char[iBufSize];

// fetch first 200 records into buffer
 iOK = GMW_DS_Fetch( iHandle, szBuf, iBufSize, 200 );
```

```
// do whatever with the data
 ODS( szBuf );

// make sure to delete the buffer
 delete [] szBuf; szBuf = NULL;

// close the query
 iOK = GMW_DS_Close( iHandle ); iHandle = 0;
 }

// Example 2:
// Get a large number of records in 100-record buffers
//
// return all serial numbers beginning with "123...."
//
char *szSQL2 = "SELECT ContSupRef, Address1, AccountNo FROM ContSupp "
"WHERE RECTYPE = 'P' AND U_CONTACT = 'SERIAL NUMBER' "
"AND U_ContSupRef Like '123%' "
"ORDER BY U_ContSupRef";
// send DataStream SQL Query
if( (iHandle = GMW_DS_Query( szSQL2 )) > 0 )
 {
 char *szBuf = NULL;
 long iBufSize = -1;
 // read while the first character of result is 0
 while( (szBuf == NULL || szBuf[0] == '0') && iBufSize )

 {
 // fetch 100 records and get the buffer size needed
 // (set the szBuf and iBufSize parameters to 0 to
 // fetch the data and retrieve the buffer size needed)
 if( iBufSize = GMW_DS_Fetch( iHandle, 0, 0, 100 ) )
 {
 // delete old buffer and allocate new buffer
 delete [] szBuf; szBuf = NULL;
 szBuf = new char[iBufSize];
 // read the data (nGetRecs is 0 since data is already read)
 iOK = GMW_DS_Fetch( iHandle, szBuf, iBufSize, 0 );
 // do whatever with the data
 ODS( szBuf );
 }
 }
 // make sure to delete the buffer
 delete [] szBuf; szBuf = NULL;
 // close the query
 iOK = GMW_DS_Close( iHandle ); iHandle = 0;
 }
 return;
}
```

Low-Level Work Area

The following example assumes that `GM5S32.DLL` has already been loaded and the function addresses have been retrieved. The example opens up the Contact1 and ContSupp tables to find a particular contact's phone number and primary e-mail address.

```
//
void DB_FuncsDLL_Example()
 {
 long iOK = 0;
 int iBufSize = 100;
 char szBuf[100], szBuf2[100], szAccNo[20+1];
//
 // Example1:
 // Find a Jon's phone number and primary e-mail address
 //
char *szName = "JON V. FERRARA";
 // open contact1 and contsupp
 long iC1 = GMW_DB_Open( "Contact1" );
 long iCS = GMW_DB_Open( "ContSupp" );
// tables opened ok?
 if( iC1 && iCS )
 {
// set the Contact1 index to ContName
 iOK = GMW_DB_SetOrder( iC1, "ContName" );
// seek Jon's name
 //
if( GMW_DB_Seek( iC1, szName ) == 1 ) // seek exact
 {
// read Jon's phone number
 iOK = GMW_DB_Read( iC1, "Phone1", szBuf, iBufSize );
 ODS( szBuf ); // show phone
 // read Jon's AccountNo
 iOK = GMW_DB_Read( iC1, "AccountNo", szAccNo, 20+1 );
 //
 // set range to all contact's e-mail records
 //
 wsprintf( szBuf, "%sPE-MAIL ADDRESS", szAccNo );
 iOK = GMW_DB_Range( iCS, szBuf, szBuf, "ContSupp" );
// loop through all e-mail records
// and find primary one
 while( iOK && (iOK = GMW_DB_Skip( iCS, 1 )) )

// read e-mail address from the ContSupRef field
 // and status from Zip
 iOK=GMW_DB_Read( iCS,"ContSupRef",szBuf,iBufSize );
 iOK=GMW_DB_Read( iCS,"Zip", szBuf2, iBufSize );

// show e-mail address
 ODS( szBuf );
```

```
// primary e-mail has a '1' in the second
// char of Zip
if( szBuf2[1] == '1' )
 break; // found primary address!
 }
 }
 // close the tables
 iOK = GMW_DB_Close( iC1 ); iC1 = 0;
 iOK = GMW_DB_Close( iCS ); iCS = 0;
 }
return;
 }{
```

Visual Basic Examples

This section contains function prototypes and examples.

Function prototypes

```
' Structure for License function
Public Type GMLicInfo
 Licensee As String * 60
 LicNo As String * 20
 SiteName As String * 20
 LicUsers As Long
 SQLUsers As Long
 GSSites As Long
 IsDemo As Long
 IsServerLic As Long
 IsRemoteLic As Long
 ISUSALic As Long
 iReserved1 As Long
 iReserved2 As Long
 iReserved3 As Long
 sReserved As String * 100
End Type
' LoadAPI Functions
Public Declare Function GMW_LoadBDE Lib "GM5S32.dll" (ByVal sSysDir As
String, ByVal sGoldDir As String, ByVal sCommonDir As String, ByVal sUser
As String, ByVal sPassword As String) As Long
Public Declare Function GMW_UnloadBDE Lib "GM5S32.dll" () As Long
Public Declare Function GMW_SetSQLUserPass Lib "GM5S32.dll" (ByVal
sUserName As String, ByVal sPassword As String) As Long
' Business logic functions
' Name-Value parameter passing to business logic function GMW_Execute(
Public Declare Function GMW_Execute Lib "GM5S32.dll" (ByVal szFunc As
String, ByVal GMPtr As Any) As Long
Public Declare Function GMW_NV_Create Lib "GM5S32.dll" () As Long
```

```
Public Declare Function GMW_NV_CreateCopy Lib "GM5S32.dll" (ByVal hgmnv As
Long) As Long
Public Declare Function GMW_NV_Delete Lib "GM5S32.dll" (ByVal hgmnv As
Long) As Long
Public Declare Function GMW_NV_Copy Lib "GM5S32.dll" (ByVal
hgmnvDestination As Long, ByVal hgmnvSource As Long) As Long
Public Declare Function GMW_GetLicenseInfo Lib "GM5S32.dll" (ByRef LicInfo
As Any) As Long
Public Declare Function GMW_NV_GetValue Lib "GM5S32.dll" (ByVal hgmnv As
Long, ByVal name As String, ByVal DefaultValue As String) As Long
Public Declare Function GMW_NV_SetValue Lib "GM5S32.dll" (ByVal hgmnv As
Long, ByVal name As String, ByVal Value As String) As Long
Public Declare Function GMW_NV_NameExists Lib "GM5S32.dll" (ByVal hgmnv As
Long, ByVal name As String) As Long
Public Declare Function GMW_NV_EraseName Lib "GM5S32.dll" (ByVal hgmnv As
Long, ByVal name As String) As Long
Public Declare Function GMW_NV_EraseAll Lib "GM5S32.dll" (ByVal hgmnv As
Long) As Long
Public Declare Function GMW_NV_Count Lib "GM5S32.dll" (ByVal hgmnv As Long)
As Long
Public Declare Function GMW_NV_GetNameFromIndex Lib "GM5S32.dll" (ByVal
hgmnv As Long, ByVal index As Long) As Long
Public Declare Function GMW_NV_GetValueFromIndex Lib "GM5S32.dll" (ByVal
hgmnv As Long, ByVal index As Long) As Long
' Low-Level DB funcs
Public Declare Function GMW_DB_Open Lib "GM5S32.dll" (ByVal sTableName As
String) As Long
Public Declare Function GMW_DB_Close Lib "GM5S32.dll" (ByVal lArea As Long)
As Long
Public Declare Function GMW_DB_Append Lib "GM5S32.dll" (ByVal lArea As
Long, ByVal sRecID As String) As Long
Public Declare Function GMW_DB_Replace Lib "GM5S32.dll" (ByVal lArea As
Long, ByVal sField As String, ByVal sData As String, ByVal iAddTo As Long)
As Long
Public Declare Function GMW_DB_Delete Lib "GM5S32.dll" (ByVal lArea As
Long) As Long
Public Declare Function GMW_DB_UnLock Lib "GM5S32.dll" (ByVal lArea As
Long) As Long
Public Declare Function GMW_DB_Read Lib "GM5S32.dll" (ByVal lArea As Long,
ByVal sField As String, ByVal sbuf As String, ByVal lbufsize As Long) As
Long
Public Declare Function GMW_DB_Top Lib "GM5S32.dll" (ByVal lArea As Long)
As Long
Public Declare Function GMW_DB_Bottom Lib "GM5S32.dll" (ByVal lArea As
Long) As Long
Public Declare Function GMW_DB_SetOrder Lib "GM5S32.dll" (ByVal lArea As
Long, ByVal Stag As String) As Long
Public Declare Function GMW_DB_Seek Lib "GM5S32.dll" (ByVal lArea As Long,
ByVal sParam As String) As Long
```

```
Public Declare Function GMW_DB_Skip Lib "GM5S32.dll" (ByVal lArea As Long,
ByVal lSkip As Long) As Long
Public Declare Function GMW_DB_Goto Lib "GM5S32.dll" (ByVal lArea As Long,
ByVal sRecNo As String) As Long
Public Declare Function GMW_DB_Move Lib "GM5S32.dll" (ByVal lArea As Long,
ByVal sCommand As String, ByVal sParam As String) As Long
Public Declare Function GMW_DB_Search Lib "GM5S32.dll" (ByVal lArea As
Long, ByVal sExpr As String, ByVal sRecID As String) As Long
Public Declare Function GMW_DB_Filter Lib "GM5S32.dll" (ByVal lArea As
Long, ByVal sFilterExpr As String) As Long
Public Declare Function GMW_DB_Range Lib "GM5S32.dll" (ByVal lArea As Long,
ByVal sMin As String, ByVal sMax As String, ByVal Stag As String) As Long
Public Declare Function GMW_DB_RecNo Lib "GM5S32.dll" (ByVal lArea As Long,
ByVal sRecID As String) As Long
Public Declare Function GMW_DB_IsSQL Lib "GM5S32.dll" (ByVal lArea As Long)
As Long
' Sync funcs
Public Declare Function GMW_NewRecID Lib "GM5S32.dll" (ByVal szRecid As
String, ByVal szUser As String) As String
Public Declare Function GMW_UpdateSyncLog Lib "GM5S32.dll" (ByVal sTable As
String, ByVal sRecID As String, ByVal sField As String, byvalsAction As
String) As Long
Public Declare Function GMW_ReadImpTLog Lib "GM5S32.dll" (ByVal sFile As
String, lDelWhenDone As Long, sStatus As String) As Long
Public Declare Function GMW_SyncStamp Lib "GM5S32.dll" (sStamp As String,
sOutBuf As String) As Long
' Datastream funcs
Public Declare Function GMW_DS_Query Lib "GM5S32.dll" (ByVal sSQL As
String, ByVal sFilter As String, ByVal sFDlm As String, ByVal sRDlm As
String) As Long
Public Declare Function GMW_DS_Range Lib "GM5S32.dll" (ByVal sTable As
String, ByVal Stag As String, ByVal sTopLimit As String, ByVal sBotLimit As
String, ByVal sFields As String, ByVal sFilter As String, ByVal sFDlm As
String, ByVal sRDlm As String) As Long
Public Declare Function GMW_DS_Fetch Lib "GM5S32.dll" (ByVal iHandle As
Long, ByVal sbuf As String, ByVal iBufSize As Long, ByVal iGetRecs As Long)
As Long
Public Declare Function GMW_DS_Close Lib "GM5S32.dll" (ByVal iHandle As
Long) As Long
Public Declare Function GMW_IsUserGroupMember Lib "GM5S32.DLL" (ByVal
szGroup As String, ByVal szUserID As String) As Long
' Misc WinAPI funcs used by VB with the GM5S32.DLL
Public Declare Sub CopyMemory Lib "kernel32" Alias "RtlMoveMemory"
(Destination As Any, Source As Any, ByVal Length As Long)
Public Declare Function lstrlen Lib "kernel32" Alias "lstrlenA" (ByVal
lpString As String) As Long
'
'
' NOTE! All GM5S32 Funcs that return a string pointer should be converted
using
' the following function. For example:
```

```
'
' sResult = PtrToStr(GMW_NV_GetValue(lGMPtr, "OutPut", ""))
'
Public Function PtrToStr(ByVal lpsz As Long) As String
Dim strOut As String
Dim lngStrLen As Long

lngStrLen = lstrlen(ByVal lpsz)
' If returning larger packets, you may have to
 ' increase this value
 lngStrLen = 64000
If (lngStrLen > 0) Then
 strOut = String$(lngStrLen, vbNullChar)
 Call CopyMemory(ByVal strOut, ByVal lpsz, lngStrLen)
 lngStrLen = lstrlen(strOut)
 PtrToStr = Left(strOut, lngStrLen)
 Else
 PtrToStr = ""
 End If
 strOut = ""

End Function
```

Logging In

```
Dim lResult As Long
lResult = GMW_LoadBDE("c:\gm5\", "c:\gm5\gmbase\", "c:\gm5\demo\",
"MASTER", "ACCESS")
If lResult <> 1 Then
 MsgBox "Unable to Load API"
```

Creating a Contact

The following example assumes that `GMXS32.DLL` has already been loaded and functions have been declared.

```
Dim lGMPtr As Long, _
 sGMnvm As String, _
 sGMvle As String, _
 lResult As Long
'//Create NV and pass pointer value to a variable
lGMPtr = GMW_NV_Create()
'//Fill Variables with Nulls
 sGMnvm = String$(100, Chr(0))
 sGMvle = String$(100, Chr(0))
'//Set Name Values
lResult = GMW_NV_SetValue(lGMPtr, "Company", "GoldMine Inc.")
lResult = GMW_NV_SetValue(lGMPtr, "Contact", "Calvin Luttrell")
lResult = GMW_NV_SetValue(lGMPtr, "Phone1", "(310)555-1212")
lResult = GMW_NV_SetValue(lGMPtr, "Email", "calvin@gm.com")
lResult = GMW_NV_SetValue(lGMPtr, "WebSite", "www.gm.com")
```

```
'//Execute Business Logic Function
lResult = GMW_Execute("WriteContact", lGMPtr)
```

Enumerating a Container

The following example assumes that GMXS32.DLL has already been loaded and functions have been declared.

```
'//Get count from NV for loop
lCount = GMW_NV_Count(lGMPtr)
For i = 1 To lCount
'//Get name from NV
 txttemp1.Text = GMW_NV_GetNameFromIndex(lGMPtr, i)
'//Get value from NV
 txttemp2.Text = GMW_NV_GetValueFromIndex(lGMPtr, i)
'//Display in list box
 sResult = txttemp1.Text + "=" + txttemp2.Text
List1.AddItem sResult
Next
```

DataStream

The following example assumes that GM5S32.DLL has already been loaded and functions have been declared.

```
sFilter = " '" + UCase$(txtMatchValue.Text) + "' $ UPPER(ContSupRef)"
iHandle = GMW_DS_Range("ContSupp", "ContSPFD", "PE-MAIL ADDRESS", "PE-MAIL
ADDRESS~", "ContSupRef;", sFilter, " ", Chr(13) + Chr(10))
If iHandle > 0 Then
 Do
 'The initial fetch will tell us how much to allocate the
 'buffer for a 50 record packet
 sBuf = String$(1, 0)
 iBufSize = GMW_DS_Fetch(iHandle, sBuf, 0, 50)
'Now, we actually grab some data…
 sBuf = String$(iBufSize + 1, 0) 'NOTE: You must initialize
 'strings to the
 'proper size before using.
 lRes = GMW_DS_Fetch(iHandle, sBuf, iBufSize, 0)

'Check if more data is available or not
 If Left(sBuf, 1) = "3" Then
 bEOF = True
 Else
 bEOF = False
 End If
'Add the results to a multi-line text box for display
 txtResults.Text = txtResults.Text + Mid(sBuf, 14, iBufSize)
Loop until bEOF
Else
 MsgBox ("Error: Invalid DS Handle!")
End If
```

Low-Level WorkArea

The following example assumes that GMXS32.DLL has already been loaded and functions have been declared. The example opens up the CONTACT1 and CONTSUPP tables to find a particular contact's phone number and primary e-mail address. The Contact name is stored in a VB Text box.

```
Dim lC1WA As Long
Dim lC2WA As Long
Dim lCSWA As Long
Dim lRes As Long
Dim sAccNo As String
Dim sBuf1 As String
Dim sBuf2 As String
'Initialization
 lblEmail.Caption = ""
 lblPrevresult.Caption = ""
 lblCompany.Caption = ""
 lblPhone.Caption = ""
 sAccNo = String$(21, 0)

 'Open data files
 lC1WA = GMW_DB_Open("Contact1")
 lC2WA = GMW_DB_Open("Contact2")
 lCSWA = GMW_DB_Open("ContSupp")
'If all files are opened OK...
 If (lC1WA And lC2WA And lCSWA) Then
'Set the index order
 Res = GMW_DB_SetOrder(lC1WA, "ContName")
'Perform the seek
 If GMW_DB_Seek(lC1WA, UCase$(txtContactName.Text)) = 1 Then
'Get the AccountNo for the matching record
 lRes = GMW_DB_Read(lC1WA, "AccountNo", sAccNo, 21)

' Get the Phone and Company fields from Contact1
'Pre-allocate string buffer
 sBuf1 = String$(100, 0)
 sBuf2 = String$(100, 0)
'Get the field data
 lRes = GMW_DB_Read(lC1WA, "Company", sBuf2, 100)
 lRes = GMW_DB_Read(lC1WA, "Phone1", sBuf1, 100)
'Update the display labels
 lblCompany.Caption = Trim(sBuf2)
 lblPhone.Caption = Trim(sBuf1)

' Get the Previous result field from Contact2
'Set the index order
 lRes = GMW_DB_SetOrder(lC2WA, "Contact2")
'Perform the seek
 If GMW_DB_Seek(lC2WA, sAccNo) = 1 Then
```

```
'Pre-allocate string buffer
 sBuf1 = String$(100, 0)
'Get the field data
 lRes = GMW_DB_Read(lC2WA, "PREVRESULT", sBuf1, 100)
'Display the field data
 lblPrevresult.Caption = sBuf1
End If
' Get the e-mail address from ContSupp
'Pre-allocate string buffer
 sBuf1 = String$(100, 0)
'Initialize the range limits
 sBuf1 = Left(sAccNo + Space$(20), 20) + "PE-MAIL ADDRESS"
'Set the range and go top
 lRes = GMW_DB_Range(lCSWA, sBuf1, sBuf1, "ContSupp")
 lRes = GMW_DB_Top(lCSWA)
'Loop until a primary e-mail is found
 Do While (lRes = 1)
'Pre-allocate string buffers
 sBuf1 = String$(100, 0)
 sBuf2 = String$(100, 0)
'Get the field data
 lRes = GMW_DB_Read(lCSWA, "ContSupRef", sBuf1, 100)
 lRes = GMW_DB_Read(lCSWA, "Zip", sBuf2, 100)
'Check if primary e-mail address
 If Mid$(sBuf2, 2, 1) = "1" Then
'Update the label
 lblEmail.Caption = Trim(sBuf1)
Exit Do 'all done
End If
'Skip to next record
 lRes = GMW_DB_Skip(lCSWA, 1)
Loop
Else
'Notify user of problem
 MsgBox ("Could not locate the specified contact.")
End If
Else
'All tables could not be opened.
 MsgBox ("Could not open the data files.")
'Exit program
 Unload Me
End If
```

Delphi Examples

This section includes function prototypes and examples.

Function prototypes

```
Type
 TGMW_LicInfo = record
 Licensee: array [0..59] of char;
 LicNo: array [0..19] of char;
 SiteName: array [0..19] of char;
 LicUsers,
 SQLUsers,
 GSSites,
 IsDemo,
 IsServerLic,
 IsRemoteLic,
 IsUSALic,
 DLLVersion,
 Reserved1,
 Reserved2:longint;
 Reserved: array [0..99] of char;
 end;

Type
 hgmnv = pointer;
// GM5S32.DLL initialization functions
function GMW_LoadBDE(sSysDir, sGoldDir, sCommonDir, sUser, sPassword:
Pchar): integer; stdcall; external 'GM5S32.DLL';
function GMW_UnloadBDE: integer; stdcall; external 'GM5S32.DLL';
function GMW_SetSQLUserPass(sUserName, sPassword: PChar):integer; stdcall;
external 'GM5S32.DLL';
function GMW_GetLicenseInfo( pGMW_LicInfo: pointer):integer; stdcall;
external 'GM5S32.DLL';
// GM5S32.DLL Sync functions
function GMW_UpdateSyncLog(sTable, sRecID, sField, cAction: PChar):integer;
stdcall; external 'GM5S32.DLL';
function GMW_ReadImpTLog(sFile: PChar; bDelWhenDone: integer; sStatus:
PChar): integer; stdcall; external 'GM5S32.DLL';
procedure GMW_NewRecID(sRecID, sUser: PChar); stdcall; external
'GM5S32.DLL';
procedure GMW_SyncStamp(sStamp, sOutBuf: PChar); stdcall; external
'GM5S32.DLL';
// GM5S32.DLL DataStream functions
function GMW_DS_Range(sTable, sTag, sTopLimit, sBotLimit, sFields, sFilter,
sFDlm, sRDlm: PChar): longint; stdcall; external 'GM5S32.DLL';
function GMW_DS_Query(sSQL, sFilter, sFDlm, sRDlm: PChar): longint;
stdcall; external 'GM5S32.DLL';
function GMW_DS_Fetch(iHandle: longint; sBuf: Pchar; iBufSize, iGetRecs:
integer): longint; stdcall; external 'GM5S32.DLL';
function GMW_DS_Close(iHandle: longint):longint; stdcall; external
'GM5S32.DLL';
```

```
// GM5S32.DLL DBF workarea functions
function GMW_DB_Open(sTable: Pchar): longint; stdcall; external
'GM5S32.DLL';
function GMW_DB_Close(lArea: Longint): longint; stdcall; external
'GM5S32.DLL';
function GMW_DB_Append(lArea: Longint; sRecID: PChar): longint; stdcall;
external 'GM5S32.DLL';
function GMW_DB_Replace(lArea: Longint; sField, sData: PChar; iAddTo:
integer): longint; stdcall; external 'GM5S32.DLL';
function GMW_DB_Delete(lArea: Longint): longint; stdcall; external
'GM5S32.DLL';
function GMW_DB_Unlock(lArea: Longint): longint; stdcall; external
'GM5S32.DLL';
function GMW_DB_Read(lArea: Longint; sField, sBuf: PChar; iBufSize:
integer): longint; stdcall; external 'GM5S32.DLL';
function GMW_DB_Top(lArea: Longint): longint; stdcall; external
'GM5S32.DLL';
function GMW_DB_Bottom(lArea: Longint): longint; stdcall; external
'GM5S32.DLL';
function GMW_DB_SetOrder(lArea: Longint; sTag: Pchar): longint; stdcall;
external 'GM5S32.DLL';
function GMW_DB_Seek(lArea: Longint; sParam: PChar): longint; stdcall;
external'GM5S32.DLL';
function GMW_DB_Skip(lArea: Longint; iSkip: integer): longint; stdcall;
external 'GM5S32.DLL';
function GMW_DB_Goto(lArea: Longint; sRecNo: PChar): longint; stdcall;
external 'GM5S32.DLL';
function GMW_DB_Move(lArea: Longint; sCommand, sParam: PChar): longint;
stdcall; external 'GM5S32.DLL';
function GMW_DB_Search(lArea: Longint; sExpr, sRecID: PChar): longint;
stdcall; external 'GM5S32.DLL';
function GMW_DB_Filter(lArea: Longint; sFilterExpr: Pchar): longint;
stdcall; external 'GM5S32.DLL';
function GMW_DB_Range(lArea: Longint; sMin, sMax, sTag: PChar): longint;
stdcall; external 'GM5S32.DLL';
function GMW_DB_RecNo(lArea: Longint; sRecID: PChar): longint; stdcall;
external 'GM5S32.DLL';
function GMW_DB_IsSQL(lArea: Longint): longint; stdcall; external
'GM5S32.DLL';
// GM5S32.DLL Quick one-field access functions
function GMW_DB_QuickSeek(sTableName, sIndex, sSeekValue, sRecID: PChar):
longint; stdcall; external 'GM5S32.DLL';
function GMW_DB_QuickRead(sTableName, sRecID, sField, sValue: Pchar; iLen:
integer): longint; stdcall; external 'GM5S32.DLL';
function GMW_DB_QuickReplace(sTableName, sRecID, sField, sValue: Pchar;
iAddTo: integer): longint; stdcall; external 'GM5S32.DLL';
// GM5S32.DLL Misc functions
function GMW_IsUserGroupMember( szGroup, szUserID: PChar): longint;
stdcall; external 'GM5S32.DLL';
```

```
function GMW_UserAccess(Option: longint): longint; stdcall; external
'GM5S32.DLL';
function GMW_CalAccess(RecType, UserID, Number1: PChar): longint; stdcall;
external 'GM5S32.DLL';
function GMW_HistAccess(RecType, UserID: PChar): longint; stdcall; external
'GM5S32.DLL';
// GM5S32.DLL business logic functions
function GMW_Execute(Func: Pchar; PGMNV: hgmnv ): longint; stdcall;
external 'GM5S32.DLL';
// create, release & copy name value containers
function GMW_NV_Create: pointer; stdcall; external 'GM5S32.DLL';
function GMW_NV_CreateCopy(PGMNV: hgmnv): pointer; stdcall; external
'GM5S32.DLL';
procedure GMW_NV_Delete(PGMNV: hgmnv); stdcall; external 'GM5S32.DLL';
procedure GMW_NV_Copy(Destination, Source: hgmnv); stdcall; external
'GM5S32.DLL';
// get and set value by name
function GMW_NV_GetValue(PGMNV: hgmnv; Name, DefaultValue: PChar): PChar;
stdcall; external 'GM5S32.DLL';
procedure GMW_NV_SetValue(PGMNV: hgmnv; Name, Value: PChar); stdcall;
external 'GM5S32.DLL';
// Check if name exists. returns: 0 failed, 1 success
function GMW_NV_NameExists(PGMNV: hgmnv; Name: PChar): longint;
stdcall;external 'GM5S32.DLL';
// remove name(s)
procedure GMW_NV_EraseName(PGMNV: hgmnv; Name: PChar); stdcall; external
'GM5S32.DLL';
procedure GMW_NV_EraseAll(PGMNV: hgmnv); stdcall; external 'GM5S32.DLL';
// iterate over name-value list (1 based)
function GMW_NV_Count(PGMNV: hgmnv): longint; stdcall; external
'GM5S32.DLL';
function GMW_NV_GetNameFromIndex(PGMNV: hgmnv; Index: longint): PChar;
stdcall; external 'GM5S32.DLL';
function GMW_NV_GetValueFromIndex(PGMNV: hgmnv; Index: longint): PChar;
stdcall; external 'GM5S32.DLL';
// Set a series of values in one shot
procedure GMW_NV_SetStr(PGMNV: hgmnv; dlmName, dlmVal: Char; StringVal:
PChar); stdcall; external 'GM5S32.DLL';
Logging In
The following example assumes that GMXS32.DLL has already been loaded and
function addresses have been retrieved
// Login to GM5
iRet := GMW_LoadBDE('C:\GM5', 'C:\GM5\GMBASE', 'C:\GM5\DEMO', 'NELSON' ,
'');
if iRet < 1 then
 ShowMessage('LoadAPI Failed. Err: '+IntToStr(iRet));
```

Creating a Contact

The following example assumes that GMXS32.DLL has already been loaded and function addresses have been retrieved.

```
// Create a new NV container
 pGMNV := GMW_NV_Create;
// Test if NV is valid
 If pGMNV <> nil then
 begin
 // Load the NVs to create the contact record
 GMW_NV_SetValue(pGMNV, 'Company', 'GoldMine Inc.');
 GMW_NV_SetValue(pGMNV, 'Contact', 'Nelson Fernandez');
 GMW_NV_SetValue(pGMNV, 'Phone1', '(310)555-1212');
 GMW_NV_SetValue(pGMNV, 'Email', 'nelson@gm.com');
 GMW_NV_SetValue(pGMNV, 'WebSite', 'www.gm.com');
// Exec the WriteContact function
 if GMW_Execute('WriteContact', pGMNV) > 0 then
 begin
 ShowMessage('Contact record was created. AccountNO=' +
 GMW_NV_GetValue(pGMNV, 'AccountNo', '') );
//Remove the pGMNV
 GMW_NV_Delete(pGMNV);
 end
 else
 // Display error
 ShowMessage('WriteContact Failed.');;
 end
else
 // Display Error
 ShowMessage('Could not create NV container.');
```

Enumerating a Container

The following example assumes that GMXS32.DLL has already been loaded and function addresses have been retrieved.

```
// Determine the number of returned values
lCount := GMW_NV_Count(pGMNV);
// If > 0 then iterate through the list
If lCount > 0 then
 For i := 1 to lCount do // Add to the results memo control
 mResults.Text := mResults.Text +
 GMW_NV_GetNameFromIndex(pGMNV,i)+'='+
 GMW_NV_GetValueFromIndex(pGMNV, i)+#13+#10;
```

DataStream

The following example assumes that GMXS32.DLL has already been loaded and function addresses have been retrieved.

```
iHandle:=GMW_DS_RANGE('Contsupp', 'Contspfd', 'PE-MAIL ADDRESS',
'PE-MAIL ADDRESS~', 'ContSupRef;', PChar('''' + UpperCase
(cebMatchValue.Text)+''' $ Upper(ContSupRef)'), '', #13+#10);
If iHandle > 0 then
Begin
 bDone :=FALSE
 Repeat
//Get Buffer Size
 iBufSize:=GMW_DS_Fetch(iHandle,NIL, 0, FETCH_SIZE);
//Allocate Buffer Memory
 pcBuffer:=AllocMem(iBufSize);
//Fetch Data
 lres:=GMW_DS_Fetch(iHandle, pcBuffer, iBufSize, 0);
 if lres>0 then //Fetch Successfully?
 begin

//Get results
 sResults:=sResults + Copy(StrPas(pcBuffer),12,iBufSize-12);
 FreeMem(pcBuffer, iBufSize); //Free buffer memory
if Copy(sHeader,1,1)<>'3' then //End of File in GM?
 bDone:=TRUE
 else
 bDone:=FALSE;
end;
until bDone
 lres:=GMW_DS_Close(iHandle);
end;
```

Low-Level Work Area

The following example assumes that GMXS32.DLL has already been loaded and function addresses have been retrieved. The example opens up the CONTACT1 and CONTSUPP tables to find a particular contact's phone number and primary e-mail address.

```
Var
 lRes, lC1WA, lC2WA, lCSWA: longint;
 aAccNo: array[0..20] of char;
 aValue1: array[0..100] of char;
 aValue2: array[0..100] of char;
begin
 // Open files
 lC1WA := GMW_DB_Open('Contact1');
 lC2WA := GMW_DB_Open('Contact2');
 lCSWA := GMW_DB_Open('Contsupp');
```

```
// Make sure all files were opened OK
 if (lC1WA>0) and (lC2WA>0) and (lCSWA>0) then
 begin
 // Set the index order
 lRes := GMW_DB_SetOrder(lC1WA, 'ContName');
// Perform the seek
 If GMW_DB_Seek(lC1WA, PChar(UpperCase(cebSearchValue.Text)) )=1 then
 begin
// Read the AccountNo
 GMW_DB_Read(lC1WA, 'AccountNo', aAccNo, 21);
// Get the field data
 lRes := GMW_DB_Read(lC1WA, 'Company', aValue1, 100);
//Display the results
 clCompany.Caption := StrPas(aValue1);
//Init the range limit string
 StrPCopy(aValue1, Copy(StrPas(aAccNo),1,20)+'PE-MAIL ADDRESS');
// Set the range and go to Top
 lRes := GMW_DB_Range(lCSWA, aValue1, aValue1, 'Contsupp');
 lRes := GMW_DB_Top(lCSWA);
// Loop through records..
 While lRes = 1 do
 begin
//Read the field data...
 lRes := GMW_DB_Read(lCSWA, 'ContSupRef', aValue1, 100);
 lRes := GMW_DB_Read(lCSWA, 'ZIP', aValue2, 100);
if aValue2[1] = '1' then
 begin
 clEmail.Caption := aValue1;
 Exit;
 end;
lRes := GMW_DB_Skip(lCSWA, 1);
end;
end
 else
 // Notify user of problem
 ShowMessage('Could not locate the specified contact!');
 end
 else
 // Notify user of problem
 ShowMEssage('Could not open all data files');
GMW_UnloadBDE;
end;
```

Resources

Additional Documentation

In addition to this guide, the following resources are available to provide you with information about GoldMine:

- **Online Help** - Accessed by clicking the Help menu option in GoldMine, online help provides topic overviews and step-by-step instructions to walk you through basic tasks, in addition to a comprehensive table of contents, index, and a search function.

NOTE: Guides are available in PDF format at: https://www.ivanti.com/support/product-documentation. Expand the GoldMine drop-down.

- **Training Courses** - Information regarding training courses for GoldMine family of products can be found at: https://www.goldmine.com/.

Contact Us

Support Site

For Support, visit: https://www.goldmine.com/goldmine-support/

Contact Information

Ivanti Software, Inc.

698 West 10000 South
South Jordan, UT 84095 USA
TEL: 1.800.443.5457

Offices are also located in Latin America, Asia Pacific, Europe, South Africa, and the Middle East. For international contact information, go to the GoldMine Web site, click the **Contact** link at the top of the page, then select your region.

GoldMine

Index

A

Activities, creating or updating 281
AddContactGrpMembers 294
AddContactGrpMembers function 293
AddFolder function 321
Alert
  attaching an alert to the specified contact record 297
  returning alerts attached to a contact record 296
  returning all defined alerts 298
API, logging in multiple users 104
Append function 44, 191
AttachTrack function 290
Automated Process
  attachingtoacontactrecord 290
  retrieving the default contact automated process 303

B

BDE session
  closing 103-104
  loading 100-101
Boolean operator 385
BR4 38
Business Logic Methods
  accessing 114
  comparing methodology to that of GM5S32.DLL 98
  using to simplify procedures 271
  working with 271

C

C++ examples for GM5S32.DLL 442
CAL.DBF 418
  SQL 417
  Xbase 399
CalComplete function 62, 208
Calendar
  completing an activity 63
  deleting Calendar items 303
CallerID function 63, 210
Close function 45, 192
code examples
  for GM5S32.DLL 454
conditionals 385
CONGRPS Structureharformat 405
CONHIST Indexesharformat 425
contact group
  adding contacts to 293
  creating 292
Contact Groups, retrieving names of 300
contact information
  accessing, using Open, Move, or Read 56, 202
  accessing, using RecordObj 56, 202
contact record
  creating or updating an additional 276, 286, 288-289
  linking contact records to an accounting application 40
Contact Record
  adding a record 132, 165
  attaching an alert to the specified contact record 297
  attaching an automated process 290
  creating or updating 271
  creating or updating a referral 280
  deleting the current record 133, 166
  evaluating an Xbase expression on a contact record 301
  reading a Contact1 or Contact2 record 295
  retrieving the default contact automated process 303
  returning alerts attached to a contact record 296
  updating notes of a primary contact record 274
CONTACT1 Relationsharformat 402
CONTACT1.DBF 422
  SQL 419
  Xbase 401
CONTACT2 Indexharformat 404
CONTACT2 Structureharformat 404
CONTACT2.DBF
  SQL 423

Xbase 404
ContactLogin function 333
ContactLogin Required NV Pairsharformat 333
CONTGRPS Structure (member records)harformat 406, 425
CONTGRPS.DBF
SQL 424
Xbase 405
CONTHIST.DBF
SQL 425
Xbase 406
CONTSUPP Indexesharformat 427
CONTSUPP.DBF 428
SQL 427
Xbase 408
COUNTER function 65, 211
CreateContactGroup function 292
CreateRemoteLicense function 309
Curtaining
checking for record curtaining 309
retrieving visible fields 308

D

data
accessing low-level data using work areas 129, 162
merging data into a document 40
retrieving data with DataStream 124, 156
data file
accessing 43, 191
closing 131, 164
opening 50, 131, 164, 195
querying for a field value 134, 167
database
file location 398
sessions, handling 271
updating information 40
database structures
CAL.DBF 418
CONTACT1.DBF 422
CONTSUPP.DBF 428
GoldMine 5.5 416
GoldMine Sales and Marketing 435
DataStream
advantages of using 125, 156
Close subcommand 67, 214
Fetch subcommand 67, 214
functions 125, 157
performance advantages 68, 216
record selection 125, 157
retrieving data with 124, 156
returning GoldMine record data 66, 213
date and time stamps
converting to TLog timestamps 86
DDE Parametersharformat 41
DDE See Dynamic Data Exchange 39
DDEINITIATE function 42
DDERequestor 38
decrypting encoded text 303
DecryptString function 303
DecryptString Required NV Pairsharformat 303
Delete function 45, 192
DeleteFolder function 321
DeleteHistory Required NV Pairsharformat 304
DeleteMail function 317
DeleteMail Required NV Pairsharformat 317
DeleteMessages functionE-mail
deleting online e-mail messages 327
DeleteMessages Required NV Pairsharformat 327, 335, 338
DeleteSchedule function 303
DeleteScheduley Required NV Pairsharformat 303
Delphi examples 448
Delphi examples for GM5S32.DLL 454
Detail Record
creating or updating 278
dialog box
displaying a message dialog box 77, 223
document link, creating or updating 76, 222
Dynamic Data Exchange 96
APPEND function 45
application service name 41
CalComplete 63
CallerID 65, 210
Counter function 65
DDE item string 41
definition 39
establishing a conversation 43
Expr function 70
Filter 47
FormAddFields function 71
FormNewFormNo 73
FormQueryCreate 74
GoldMine license macros 96, 244
GoldMines DDE server 43
identifying incoming telephone numbers 40
inserting incoming e-mail 40
InsHistory 75
LinkDoc 77

linking e-mail to external systems 40
macros 86, 234
merge form macros 95, 244
merging a document with 40
Move 50
MsgBox 79
MsgBox function 78, 224
NewForm 81
NewGroup 82
RecNo 53
Replace function 54
Search 55
SendPage 85
service topic 41
StatusMsg 86
transferring data to accounting application 40
updating database 40
using to query for data 40
working with DDE functions 43

E

E-mail
accessing e-mail templates 322
account information, retrieving 323
adding an PlaceNameE-mail PlaceTypeCenter folder 321
deleting an PlaceNameE-Mail PlaceTypeCenter folder 321
filing a message in History 317
managing internet e-mail preferences 328
name/value functions 311
obtaining a list of PlaceNameE-Mail PlaceTypeCenter folders 321
queuing a message for delivery 316
retrieving a manual list of recipients 328
retrieving e-mail account information 323
returning a list of unique From addresses 322
saving a manual list of recipients 328
updating an e-mail address 273
empty child container, creating 119
empty record
adding 44, 191
encrypting text 302
EncryptString function 302
EncryptString Required NV Pairsharformat 302
exported records
counting the number of 74, 220
Expr function 69, 217
external application
linking with GoldMine fields 56, 201

F

field
returning a FormNo value to register unattached fields 73
field name
returning for an expression, macro, or field 73
field value
changing 54, 135, 168
querying a data file for 134, 167
reading 143, 178
replacing 144, 178
FieldAccessRights function 308
FieldAccessRights Required NV Pairsharformat 308
FileMail function 317
FileMail Required NV Pairsharformat 318
filter creation 136, 169
Filter function 45, 193
FolderList function 321
form
adding merge fields 71
closing a profile 72
FormAddFields function 70, 217
FormAddFields function See Dynamic Data Exchange 71
FormClearFields function 71, 218
FormCloseForm function 72, 218
FormCreateFile function 72, 218
FormGetFieldName function 73, 219
FormNewFormNo function 73, 220
FormQueryCreate function 74, 220
FromList function 322

G

GetAccountsList function 323
GetActiveOppty function 62, 208
GetAllAlerts function 298
GetContactAlerts function 296
GetEmailPrefs function 328
GetGroupName function 300
GetGroupUsersList function 299
GetLoginCredentialsfunction 61, 206-207
GetManualRcptList function 328
GetNewContactAP function 303
GetUserAccess function 306
GetUserMemberships function 299
GetUsersList function 298

GM5S32.DLL 129, 162
database access and sync log updates 97
loading and logging in 98
synchronization functions 145, 181
GM5S32.DLL code examples 454
C++ 442
Delphi 454
Visual Basic 448
GM5S32.DLL Low-Level Access Functionsharformat 130, 163
GM5TP.DLL 107
GMW_DB_Append function 132, 165
GMW_DB_Bottom function 142, 176
GMW_DB_Close function 131, 164
GMW_DB_Close Return Valuesharformat 132
GMW_DB_Delete function 133, 166
GMW_DB_Filter function 136, 169
GMW_DB_Filter Return Valuesharformat 136, 170
GMW_DB_Goto function 140, 174
GMW_DB_IsSQL function 132, 165
GMW_DB_Move Commands and Function Equivalentsharformat 139, 173
GMW_DB_Move function 139, 173
GMW_DB_Open function 131, 164
GMW_DB_QuickRead function 143, 178
GMW_DB_QuickReplace function 144, 178
GMW_DB_QuickSeek function 142, 177
GMW_DB_Range function 136, 170
GMW_DB_Read function 134, 167
GMW_DB_RecNo function 134, 167
GMW_DB_Replace function 135, 168
GMW_DB_Search function 137, 171
GMW_DB_Seek function 138, 171
GMW_DB_SetOrder function 138, 172
GMW_DB_Skip function 141, 175
GMW_DB_Skip Return Valuesharformat 142, 175
GMW_DB_Top function 141, 175
GMW_DB_Unlock 135, 169
GMW_DS_Close 125, 129, 157, 162
GMW_DS_Fetch 125, 157
GMW_DS_Query 125, 157
GMW_DS_Range 125, 157
GMW_Execute function 114
GMW_GetLicenseInfo function 123-124
GMW_LoadBDE function 100-101, 150, 152-153
GMW_MUBeginSession function 106
GMW_MULogin function 105
GMW_MULogout function 106, 154
GMW_NewRecID function 147, 183
GMW_NV_AppendEmptyNvValue function 120, 332
GMW_NV_AppendNvValue function 332
GMW_NV_AppendValue function 119-120
GMW_NV_Copy function 108
GMW_NV_Count function 112
GMW_NV_Create function 107
GMW_NV_CreateCopy function 108
GMW_NV_Delete function 109
GMW_NV_EraseAll function 111
GMw_NV_EraseName function 118
GMW_NV_EraseName function 111
GMW_NV_GetMultiValue function 118
GMW_NV_GetMultiValueCount function 116
GMW_NV_GetNameFromIndex function 112
GMW_NV_GetNVValue function 117
GMW_NV_GetValue function 109
GMW_NV_GetValueFromIndex function 113
GMW_NV_GetValueType function 114
GMW_NV_IsRoot function 115
GMW_NV_NameExists function 110
GMW_NV_SetNvValue function 118
GMW_NV_SetStr function 113
GMW_NV_SetValue function 110
GMW_ReadImpTLog 230
GMW_ReadImpTLog function 146, 182, 230
GMW_ReadImpTLogharformat 146, 182
GMW_SetSQLUserPass function 99
GMW_SyncStamp function 148, 183
GMW_UnloadBDE function 103-104
GMW_UpdateSyncLog function 145, 181, 229
GMW_UserAccess function 121, 231
GoldMine 5.5 database structures 416
GoldMine KnowledgeBase 37
GoldMine license macros see Dynamic Data Exchange 96, 244
GoldMine Sales and Marketing database structures 435
group
adding a group member 82
creating an empty group 81

H

History
filing a message in History 317
history record
creating 74, 221
creating or updating 285

I

IIS extensions, and multi-threaded applications 107

import file
importing a prepare TLog import file 146, 182, 230
index
setting the current index tag 138, 172
INFOMINE.DBF
SQL 429
Xbase 410
InsHistory function 74, 221
InsHistory Valid Values (2nd parameter)charformat 74, 221
integrating with GoldMine
methods 34
integration tools
DDERequestor 38
interfacing with GoldMine 398, 417
internet
e-mail preferences 328
IsContactCurtained function 309
IsSQL function 47, 193

K

KnowledgeBase 37

L

license
generating a remote license file 309
removing a remote license 310
returning GoldMines Licensing Information 123-124
LinkDoc function 76, 222
linked document
creating or updating 279
logical evaluators 385
logicals 388
login
creating a new GoldMine login 305
login sessions, switching between 106
LOOKUP Indexesharformat 411
LOOKUP.DBF
Xbase 411

M

macro
identifying by file name 83, 226
identifying by number 83, 226
macros 83, 226
creating 83, 226
DDE macros for Merge Forms 93, 242
DDE macros for the GoldMine License 95, 244
mail message
deleting a message 317
deleting online e-mail messages 327
filing a message in History 317
preparing an Name/Value container to forward a mail message 320
preparing the NV container for a new mail message 318
queuing a message for delivery 316
reading 311
retrieving a list of messages waiting online 324
retrieving online messages 326
saving a mail message into GoldMine 316
updating 316
MAILBOX Indexesharformat 430
MAILBOX.DBF
SQL 430
Xbase 411
merge fields added to a form 71
merge form
adding 79, 225
DDE macros See Dynamic Data Exchange 95, 244
merging data into a document 40
message dialog box display 79
message,displaying in GoldMines status bar 85, 228
Move function 47, 194
mrecord
moving to the previous or following record 141
MS Word for Windows, Linking GoldMine to 41
MsgBox function 77, 223
multi-threaded applications
special considerations 106
multi-value NV pairs 116
appending string values 120
deleting values from 118
retrieving values 118

N

Name/Value container
assigning a container to a parent 118
copying values between containers 108
creating 107
creating an empty child container within the parent 119
creating with copied values 108
deleting a container 109
determining container position in NV hierarchy 115
preparing an NV container to forward a mail message 320
preparing the container for a new mail message 318
reading values from a container 109

retrieving containers from an NV pair 117
storing NV pairs in a container 110
Name/Value Functions 107
E-mail 311
Name/Value pair
determining the type of an NV pair 114
finding an NV name 112
finding an NV value 113
getting the number of values in a multi-value pair 116
removing all NV pairs from a container 111
removing one NV pair 111
retrieving values in a multi-value pair 118
searching for an NV pair 110
setting NV pairs 113
totaling NV pairs in a container 112
working with multi-value NV pairs 114
NewForm function 79, 225
NewGroup function 81
NewMember function 82
NonCurtainedFields function 308
Notes, updating notes of a primary contact record 274

O

OnlineList function 324
Open function 50, 195
operators 385
OPMGR Indexesharformat 432
OPMGR.DBF
SQL 432
Xbase 413

P

pager message
creating and sending 84, 228
PERPHONE Indexesharformat 433
PERPHONE.DBF
SQL 433
Xbase 414
placeCityWriteContactOutput StateNVharformat 272
PlayMacro function 83, 226
PrepareFwdMail Required NV Pairsharformat 320
PrepareNewMail function 318

Q

QueueMail function 316
QuickRead 143, 178
QuickReplace 144, 178
QuickSeek 142, 177

R

Range function 51, 196
Read function 52, 197
ReadContact function 295
ReadMail function 311
ReadRecord function 294
ReadRecord Required NV Pairsharformat 294
RecNo function 52, 198
record
checking the current record number or record ID 134, 167
creating a subset of records 136, 169
deleting the current record 45, 192
getting a new record 183
moving to a specified record 47, 140, 174, 194
moving to the first match 138, 171
moving to the first record 141, 175
moving to the last record 142, 176
moving to the previous or following record 175
positioning the pointer to a specified record 139, 173
reading a 294
unlocking 56
unlocking a record 135, 169
RecordObj
subfunctions 57, 202
RecordObj function 56, 201
referral, creating or updating 280
remote license
generating a remote license file 309
removing 310
RemoveRemoteLicense function 310
Replace function 53, 199
RESITEMS.DBF
SQL 434
Xbase 415
RetrieveMessages function 326

S

SaveMail function 316
SaveManualRcptList function 328
search
limiting the search scope 136, 170
performing a sequential search 137, 171
SEARCH function 54, 200
Security
handling GoldMine Security 305
reading security and rights for a DLL user 121, 231
retrieving field-level access rights 308

retrieving security access 306
validating a Web user name and password 333
seek
moving to the first record match 138, 171
seeking a record 142, 177
SendPage function 84, 228
service item 86, 234
service name 41
service topics 41
SetContactAlert function 297
SetEmailPrefs and CityplaceGetEmailPrefs StateNV Pairsharformat 329
SetEmailPrefs function 328
SetSessionHandling function 271
SPFILES.DBF
SQL 434
Xbase 416
SQL
determining whether a table is SQL or Xbase 132, 165
executing a query 291
setting the database login name and password 99
table, checking for 47, 193
SQL database structures 435
SQLStream function 291
status bar
message display 86
StatusMsg function 85
Summary tab 92, 240
support and resources
GoldMine KnowledgeBase 37
sync log
updating sync logs with GM5S32.DLL 145, 181
updating the Sync Log file 145, 181
sync stamp
converting to time format 148, 183
synchronization functions 145, 181
SyncStamp function 86, 228
System Agent 227

T

table
checking for an Xbase or SQL table type 47, 193
moving to the last record 142, 176
TemplateList function 322
templates, accessing e-mail templates 322
third-party developers 398, 417
timestamps
converting TLog 86, 228
TLog import file
importing a prepared TLog import file 146, 182, 230
TLog Import Structureharformat 147, 183, 230
TLog timestamps
converting to date and time stamps 86

U

UNLOCK function 201
UpdateEmailAddress function 273
UpdateMail function 316
UpdateWebSite function 274
user
creating a new GoldMine login 305
generating a remote license file 309
logging in multiple users through the API 104
reading security and rights for a DLL User 121, 231
removing a remote license 310
retrieving field-level access rights 308
retrieving security access 306
returning a user list 298
returning group memberships for a specified user 299
validating a Web user name and password 333
user group
returning a user group member list 299
returning group memberships for a specified User 299
saving a user group 300

V

VBA 42-43
visible fields, retrieving 308
Visual Basic examples for GM5S32.DLL 448
Visual Basic for Applications 42

W

Web
validating a Web user name and password 333
Web import instruction file, processing 216
Web site record,updating 274
Work Area
accessing low-level data using work areas 129, 162
in DDE functionsharformat 42
WriteContact function 271
WriteContact Special NV Pairsharformat 272
WriteContactNotes function 274
WriteDetail function 278
WriteGMUser function 305
WriteGroupUsersList function 300
WriteHistory function 285

WriteLinkedDoc function 279
WriteLinkedDoc Optional NV Pairsharformat 279
WriteOtherContact function 276, 286, 288-289
WriteOtherContact Special NV Pairsharformat 277
WriteReferral function 280
WriteSchedule function 281

X

Xbase
- conditionals, operators, and logical evaluators 385
- creating an Xbase file with registered fields 72, 218
- date functions 388, 392
- determining whether a table is SQL or Xbase 132, 165
- evaluating an Xbase expression on a contact record 301
- expression, reading without opening a file 69, 217
- function/parameter types 384
- functions 388
- miscellaneous functions 388, 396
- numeric functions 388, 394
- string functions 388-389
- table, checking for 47, 193

Xbase database structures 416
XbaseContactExpr function 301
XbaseContractExpr Return NV Pairsharformat 301