Table of Contents:

DOC1: GM4's New Sync

This article describes the new GoldMine 4.0 low-level synchronization method and its sync transaction log. The GoldMine 4.0 synchronization is very simple because all changes of all fields from all tables are tracked in the exact same way.

There is a sync transaction log table (a TLog table) for all tables in the GoldMine directory named GMTLog, and a TLog table named ContTLog for all tables in a contact set directory. Records in GoldMine 4.0 are uniquely identified by a RecID (what used to be TransNo). When new records are created, they are assigned a RecID which is guaranteed to be unique, even across multiple GoldMine systems. The RecID field is always the last field in a table, and the index on the RecID field is always the last tag of the table's indexes. Entries in the TLog tables are uniquely identified by the RecID and FieldName, thereby tracking updates of each field in each record of each table in GoldMine.

The transaction log record (the TLog record) consists of four fields; The LogStamp field contains a compressed value of the current date and time, down to the millisecond. The Action field contains one of following three values; "N" -- a new record has been created, "D" -- the record has been deleted, and "U" -- a field has been updated. The FRecID field contains the record's RecID, prefixed by the TableID + 33. The last Field, FieldName contains the updated field's name, or "zzNew" for new records which have never been synced, or "zsNew" for new records which have been synced at least once. FieldName contains "zzzDel" for deleted record.

When a field is changed, GoldMine first searches for the record's transaction log record (the TLog record) with a "zzNew" entry in FieldName. If this record is found, it means that the record has never been synced since it was created, so there is no reason to track the changes of each field individually. When the record is first synced, the "zzNew" TLog entry is changed to "zsNew" to indicate that the record has been synced. From that point on, the record's field changes are tracked individually by creating a separate TLog record for each changed field in the record. Since most changes of new records are made shortly after the record has been created, this mechanism greatly minimizes the number of TLog entries per record.

DOC2: GM4 now Multithreaded !!

GoldMine 4.0 uses Win32 threads to execute background processes, allowing for much better performance and responsiveness to the user. All GoldMine background processes are centrally controlled from the new Process Monitor window.

GoldMine 3.2 and prior versions multitask using "cooperative" multitasking methods. This means that while "background" processes like Automated Processes and Global Replace are running in GoldMine 3.2, the two processes are actually running in the same thread, cooperatively yielding to each other and sharing the same database cursors. 90% of all Windows applications work this way.

The "cooperative" multitasking method presents two performance problems; first, the two processes have to save their cursors state prior to yielding control, and restoring the cursors after receiving control back. Second, since the User Interface (the UI) and all "background" processes execute in the same primary thread, the Win32 operating system cannot identify the UI code and make it more responsive to the user.

GoldMine 4.0 now runs its processes in their own threads, using their own private cursors. This offers three major performance gains. When the user interacts with GoldMine while background processes are running, the Win32 operating system will immediately reduce the priority of the background processes and schedule high-priority CPU time to the UI thread to make it very responsive to the user. As the user pauses their clicking and typing, Win32 schedules more CPU time for the threads running the background processes. Windows does a great job at this.

Since each GoldMine 4.0 threaded background process uses its own cursors, there is no longer a need to save and restore the state of the cursors, which allows the processes to execute faster. In addition, when running GoldMine 4.0 on a multiprocessor Windows NT 4.0 system, the NT operating system balances the GoldMine 4.0 multithreaded processes very effectively to utilize the computer's multiprocessors to their fullest.

All GoldMine 4.0 background processes can be monitored from the new "Process Monitor" window, a.k.a. ProcMon. Background processes include Automated Processes, Global Replace, Synchronization, Mail Merge, Reports, Auto Get/Send Email, Building Groups, Group Scheduling, Global Delete, Importing and Exporting data.

GoldMine 4.0 can run dozens of background processes simultaneously. You can even run multiple instances of the same process, like multiple Automated Processes scans, Global Replace on multiple filtered sets and building multiple groups, all at the same time. The GoldSync and GoldMine synchronization status is also recorded in ProcMon.

One of the more interesting multiprocessing tests involves direct Internet IP to IP synchronization where both sides have two processes running, a listener and a caller. When both callers call together, both listeners on each machine will answer and perform two independent and simultaneous syncs. During the sync, each of the 4 tasks (2 callers and 2 listeners) perform 2 simultaneous actions of creating the transfer set for the caller while downloading the set sent by the caller, all and all 8 simultaneous threads, 4 on each system. Starting additional processes like an Automate Processes scan and a group Mail Merge during the sync should hardly effect the performance of active synchronization processes.

And one final performance note; Creating a sync transfer set of 1500 contact1 records from REX (my NT4 server) takes about 40 seconds to create, 80 seconds to upload the 200K file over the Internet, and another 60 seconds to be retrieved by the remote GoldMine. This 3 minute, 1500 record Internet sync is about 3 times faster than GoldMine 3.2 syncs. Normal daily syncs of 300 records should take about a minute. Incidentally, the new Internet IP to IP direct synchronization method is the absolute fastest way to synchronize GoldMine data (and it has been the most robust sync method throughout alpha testing). Finally, any standard GoldMine sync process can be started to periodically sync with a "home" GoldSync system. This allows remote users to set up automatic background synchronization throughout the day.

DOC3: GM4's DDE Server

This documents describes the new DDE features available in GoldMine 4.0's DDE server.

The GoldMine 4.0 DDE server is syntactically compatible with GoldMine 3.2 and prior versions. There are some differences however, which may require existing GoldMine DDE code to be updated. Since SQL databases are strict about their field names, there are a few field names which have changed between GM32 and GM4, most notably is the USER field which is now named USERID. The index offsets into SETORDER have also been changed to accommodate the new indexes in GoldMine 4.0. A listing of field name changes, index offsets and tables IDs is posted in a separate document. Converting GM32 DDE code to GM4 should be relatively simple.

DDE WORKAREAS:
==========
Internally, the GoldMine 4.0 DDE server behaves slightly differently than GoldMine 3.2 and prior versions. The DDE low-level commands (OPEN / APPEND / REPLACE / CLOSE, etc...) now open up their own independent thread-safe database cursors. This speeds up the DDE access since the table state no longer has to be saved and restored during each low-level DDE call (as it does in GM32). This feature also allows for opening the same table multiple times, allowing multiple independent navigational cursors on the same table. In addition, all GoldMine tables are now supported by the low-level DDE commands, not just the primary tables as with GM32. All updates to all tables will synchronize.

Opening private DDE cursors has one drawback compared to GM32. If an OPENed file is not CLOSEed, the allocated cursor is not released, causing a significant resource leak. This can also cause "Table is Busy" errors when the table needs to be opened exclusively (during re-indexing, for example). On SQL tables, the unclosed cursors will keep the connection needlessly active, which is very undesirable on SQL servers.

Since not all developers write bug-free code, when a developer does not CLOSE an OPENed cursor, GoldMine will try to automatically close all opened DDE cursors when the last DDE conversation is terminated. However, if the client application which opened the cursor crashes without closing the cursor or terminating the conversation, the cursor will be left open. When GoldMine terminates, it will again scan all DDE allocated cursors and close any of which are left open. During beta, you will see a "DEBUG: Shutdown: 1 DB_Table still active!!!!!" message when GM4 terminates and cursors are remained open.

When opening multiple cursors of the SAME table, a conflict can arise when the two workareas of the same table try to evaluate a dBASE expression (using macros or

Expr). Since macro expressions are evaluated against the current state of Contact1 in the Record window, or whichever table the expression is aliasing, the evaluations are performed against the cursors of the Record window and not the workarea cursor. You can, however, evaluate an expression against the state of the workarea's cursor by passing a valid expression into the READ command.

The FILTER & RANGE Commands:
==========
GoldMine 4.0 includes two powerful new low-level DDE command, FILTER and RANGE. The FILTER command takes the workarea as param 1 and an xBase filter expression as param2. The FILTER command works just like the dBASE/Clipper filter commands do, in the way in which it filters in only the records evaluating to a TRUE condition. Different filters can be applied for each workarea. To cancel the filter, call FILTER with an empty string.

The FILTER command can be very useful on smaller tables, or on table ranges, but it could be very slow on large tables. One way to manually achieve ranged filters is by SEEKing to the first record and then applying the filter. For example, the following code fragment loops through all history records of the current contact having the word "SALE" in the reference field:

```
OPEN( conthist )
SEEK( wa, sTheAccNo, index )
FILTER( wa, " 'SALE' $ conthist->ref' " )
DO WHILE( READ( wa, accountno ) = sTheAccNo )
   // do your thing...
CLOSE( wa )
```

In addition to the FILTER command, the RANGE command limits a cursor obtained by OPEN to a range of records ordered by an index. Using a ranged cursor, a simple go TOP, SKIP while not EOF loop can fetch all the ranged records. Another advantage of the RANGE command is that it issues a single optimized SQL query to fetch the records, resulting in the fastest data retrieval from SQL databases. The RANGE command uses the following syntax:

```
RANGE( workarea, min, max, indextag )
```

The following two examples show how to limit the contact1 cursor to all contacts in the 90xxx zip code, and how to limit a ContSupp cursor to the profile records of a given contact:

```
RANGE( waC1, "90000", "90999", ContZip )
RANGE( waCS, sC1AccNo+'P', sC1AccNo+'P', ContSupp )
```

To cancel a range, call RANGE with empty strings for min, max, and tag, like

```
RANGE( wa, "", "", "" )
```

A new range can be issued without canceling the old range since GoldMine will cancel the current range before applying a new range. You can apply a FILTER to a range. You should NOT use the MOVE SEEK function with a range. Following is an example that uses FILTER and RANGE to fetch all tech support calls from the contact's history. This code fragment demonstrates the fastest possible way to retrieve the data, both from dBASE and SQL databases:

```
OPEN( ContHist )
RANGE( wa, sAccNo, sAccNo,  ContHist )
FILTER( wa, "left(conthist->actvcode,2) = 'TS' " )
MOVE( wa TOP )
```

```
DO WHILE NOT EOF
   // do your thing....
   MOVE( wa, SKIP )
CLOSE( wa )
```

DDE MONITOR:
============
Have you ever wanted to see how your DDE commands are received by GoldMine, and what GoldMine's results are? Well... you can now monitor all GoldMine's DDE conversations, their input commands and output results, directly on the new Process Monitor. In addition, ProcMon's log section lists your DDE commands in the same order in which they are sent by your program.

To turn DDE Monitor on, send in a [DDEMonitor] command. To turn it off, send a [DDEMonitor(0)] command. For example, using the DDEReq32 utility, send the following commands:

```
[DDEMontor]
&Contact1
[FormCreateFile("100123","d:\test",5)]
```

The FormCreateFile command will build a merge file, showing two processes in the Process Monitor, one for the DDE Monitor and one showing the mail merge status (please substitute the 100123 FormNo with a FormNo value from your FormsFld file).

UPDATING SYNC LOGS VIA DDE
==========================
GoldMine 4.0 allows developers to open and manipulate the GoldMine tables directly, using a wide variety of development tools. When GoldMine data is updated externally, the external application can send an UpdateSyncLog DDE command to update GoldMine's sync logs to reflect the changes made externally. The UpdateSyncLog command uses the following syntax:

```
UpdateSyncLog( Table, RecID, FieldName, Action )
```

The 'Table' parameter is the Table name (like "Contact1") or the TableID. The 'RecID' parameter is the RecID of the updated record. It is imperative that the correct RecID is passed, and that it is exactly 15 characters long. The 'FieldName' parameter is the field name which has been changed. This parameter is only relevant when the Action parameter is 'U'. It is ignored when Action is 'N' or 'D'. The 'Action' parameter 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.

The possible return values from UpdateSyncLog are as follows:

```
 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
```

When updating multiple fields of the same record, UpdateSyncLog must be called for each modified field. However, if the first call returns 1 or 2, it is not necessary to call UpdateSyncLog for the rest of the modified fields. The following example updates the sync TLog to indicate that the Key1 field of the record identified by a "^%J9$0%&X;$Z)XD" RECID has been changed:

```
   [UpdateSyncLog("contact1",  "^%J9$0%&X;$Z)XD", "Key1", "U")]
```

The GoldMine 4.0 synchronization internals, RecIDs and TLogs will be covered in detail in documents to follow.

DOC4: GM4 Data Structure

This document details the GoldMine 4.0 tables and indexes, and the field name changes between GoldMine 3.x/2.x and GoldMine 4.0.

The field definitions are now stored in a table named DATADICT.DBF. This used to be SPDBFS.DBF in GoldMine 3.x/2.x. You should not modify the DATADICT file.

The following table lists all GoldMine 4.0 database tables.
The TID code represents the table ID in the TableID field of the TLogs.

ID	TID	Table	Name
0	!	LookUp	LookUp
1	"	Cal	Calendar
2	#	Forms	Forms
3	$	GMTLog	Primary TLogs
4	%	ContTLog	Contact Files TLogs
5	&	Contact1	Contact1
6	'	Contact2	Contact2
7	(	ContSupp	Contact Supp
8	)	ContHist	Contact History
9	*	ContGrps	Contact Groups
10	+	Filters	Filters
11	,	DataDict	Data Dictionary
12	-	SpFiles	Database Directory
13	.	Users	Users
14	/	PerPhone	Rolodex
15	0	ContUDef	User Defined Fields
16	1	Resource	Resource
17	2	ImpExp	Import/Export/Purge
18	3	InfoMine	InfoCenter
19	4	ScriptsW	Scripts
20	5	Report32	Reports
21	6	Fields	Fields
22	7	Tracks	Tracks
23	8	FormsFld	Forms Field
24	9	UserLog	User Logs
25	:	Reports	Reports (Crystals)
26	;	MailBox	MailBox
27	<	ErrorLog	Error Logs
28	=	SyncLog	Sync Logs
29	>	SyncProc	Sync Processes
30	?	SyncSite	Sync Sites
31	@	SyncTask	Sync Tasks
32	A	SyncLock	Sync Lock File
33	B	OpMgr	Opportunity Manager
34	C	LeadDbfs	Leads Directory
35	D	LeadFile	Leads Analysis File

The following table lists the indexes used by GoldMine 4.0. The TAG ID is the offset number into the DDE SETORDER and RANGE commands.

Tag ID	Table	Index	Index Key
0	LookUp	LOOKUP	fieldName+entry
1	LookUp	LKURECID	recId
2	Cal	CAL	rectype+userId+onDate+onTime
3	Cal	CALCONT	accountNo+rectype+onDate+onTime
4	Cal	CALDATE	userId+onDate+onTime
5	Cal	CALPROB	rectype+userId
6	Cal	CALALARM	alarmFlag+userId+alarmDate+alarmTime
7	Cal	CALRLINK	lopRecId
8	Cal	CALRECID	recId
9	Forms	FORMS	recType+userId
10	Forms	FRMRECID	recId
11	GMTLog	GTLOGDAT	tableId+syncStamp
12	GMTLog	GTLOGTRN	tableId+fRecId+fieldName
13	ContTLog	CTLOGDAT	tableId+syncStamp
14	ContTLog	CTLOGTRN	tableId+fRecId+fieldName
15	Contact1	CONTACC	accountNo
16	Contact1	CONTCOMP	company+accountNo
17	Contact1	CONTNAME	contact+accountNo
18	Contact1	CONTZIP	zip+accountNo
19	Contact1	CONTCITY	city+accountNo
20	Contact1	CONTkey1	key1+accountNo
21	Contact1	CONTkey2	key2+accountNo
22	Contact1	CONTkey3	key3+accountNo
23	Contact1	CONTkey4	key4+accountNo
24	Contact1	CONTkey5	key5+accountNo
25	Contact1	CONTLAST	lastName+accountNo
26	Contact1	CONTPHON	phone1+accountNo
27	Contact1	CN1RECID	recId
28	Contact2	CONTACT2	accountNo
29	Contact2	CN2RECID	recId
30	ContSupp	CONTSUPP	accountNo+recType+contact
31	ContSupp	CONTSPFD	recType+contact+contSupRef
32	ContSupp	CNSRECID	recId
33	ContHist	CONTHIST	accountNo+onDate
34	ContHist	CONTHUSR	userId+sRecType+onDate
35	ContHist	CNHRLINK	lopRecId
36	ContHist	CNHRECID	recId
37	ContGrps	GROUPNO	userId+code
38	ContGrps	GROUPACC	accountNo+userId
39	ContGrps	GRPRECID	recId
40	Filters	FILTERS	recType+userId+name
41	Filters	FLTRECID	recId
42	DataDict	DATADICT	dbfName+field_name
43	SpFiles	SPFILES	DirPath
44	SpFiles	SFLCODE	DirCode
45	SpFiles	SFLRECID	recId
46	Users	USERS	userName
47	Users	USERGRUP	userGroup
48	Users	USRRECID	recId
49	PerPhone	PERPHONE	recType+userId+contact
50	PerPhone	PPHRECID	recId
51	ContUDef	CONTUDEF	dbfName+field_name

52	ContUDef	CNURECID	recId
53	Resource	RESOURCE	name
54	Resource	RSCRECID	recId
55	ImpExp	IMPEXP	recType+number+entryDesc
56	ImpExp	IMPRECID	recId
57	InfoMine	INFOMINE	recType+tsection+topic
58	InfoMine	INFOSORT	sortKey
59	InfoMine	INFOTRAN	recType+recId
60	InfoMine	INFRECID	recId
61	ScriptsW	SCRIPTSW	recType+questNo+ansNo
62	ScriptsW	SCRPcode	recType+ansNo
63	ScriptsW	SRTRECID	recId
64	Report32	REPORT32	recType+userId+repDesc
65	Report32	REPRECID	recId
66	Fields	FIELDS	recType+viewId+fldPos+col+row
67	Fields	FLDRECID	recId
68	Tracks	TRACKS	recType+trackNo+eventNo
69	Tracks	TRKCODE	recType+eventNo
70	Tracks	TRKRECID	recId
71	FormsFld	FORMSFLD	formNo+recType+Label
72	FormsFld	FFMRECID	recId
73	UserLog	USERLOG	userId+login
74	UserLog	ULGRECID	recId
75	Reports	REPORTS	recType+userId+repDesc
76	Reports	R16RECID	recId
77	MailBox	MBOXLINK	LinkRecId
78	MailBox	MBOXUSER	userId+folder
79	MailBox	MBXRECID	recId
80	ErrorLog	ERRORLOG	onDate
81	ErrorLog	ERRRECID	recId
82	SyncLog	SLCATG	category+ldate+ltime
83	SyncLog	SLSITE	category+siteId+ldate+ltime
84	SyncLog	SLMACHID	category+machineId+processId+ldate+ltime
85	SyncLog	SLRECID	recId
86	SyncProc	SPMACHID	machineId+processId
87	SyncProc	SPRECID	recId
88	SyncSite	SSID	siteId+siteName
89	SyncSite	SSSECT	sSection+siteId
90	SyncSite	SSQ	recType+nextSync
91	SyncSite	SSRECID	recId
92	SyncTask	STID	siteId
93	SyncTask	STWRING	wanRing
94	SyncTask	STRECID	recId
95	OpMgr	OPMGR	recType+userId+stage
96	OpMgr	OPID	opId+recType
97	OpMgr	OPRECID	recId
98	LeadDbfs	LDDESC	fileDesc
99	LeadDbfs	LDRECID	recId
100	LeadFile	LFSOURCE	recType+source
101	LeadFile	LFSORT	recType+sort
102	LeadFile	LFRECID	recId

The following table lists all field name changes between GoldMine 3.x/2.x and GoldMine 4.0:

NEW FIELD	OLD FIELD
// all tables	

```
"UserID"          "User"
"LastTime"        "LastTime".to24hr
"CreateAt"        "CreateAt".to24hr
"RecID"     "TransNo"

// lookup
"FieldName"       "FieldName"+"Status"
"Entry"           "Desc"

// cal
"OnDate"          "Date"
"OnTime"    "Time"

// conthist
"SRecType"        left("RecType",1)
"OnTime"    if(RecType[0] == 'A', Secs2Time(RecType[1]*10*60), LastTime.to24hr)

// contgrps
"RecID"     left("AccountNo" 8)

// filters
"RecType"   "F"
"QExpr"     "FilterExp"

// tracks
"TriggerTyp"      "Trigger"

// infomine
"TSection"        "Section"

// fields
"FldPos"    "Position"

// forms
"FormDesc"        "Desc"

// impexp
"EntryDesc"       "Desc"
"FilterExpr"      "Filter"

// report32
"RepDesc"   "Desc"

// Resources
"ResDesc"   "Desc"

// users
"UserGroup"       "Group"
"USecurity"       security fields

// mailbox
Cal and ContHist Internet email records
are moved to the MailBox table
```

DOC5: Sorting Browse Columns

This documents describe the new browse sorting feature available in GoldMine 4.0.

GoldMine 4.0 allows users to sort the records in browse windows by clicking on their column titles. When the header of the sort columns is depressed, that column is sorted by Ascending order. When the column header appears raised, the column is sorted by Descending order. The painting here will be cleared up in a future build.

Sorting columns works only on those columns which show a single field from the database. Columns where GoldMine has to derive the displayed result cannot be sorted since there's no (practical) way to convert all that logic to SQL. For example, in the Groups browse window, the members column shows a value which is compressed into the group header record, and thus cannot be used for issuing an SQL query.

When GoldMine 4.0 sorts by a column, it sends an SQL query for the ordered the range of records. The column sorting feature was originally available only for SQL databases, since issuing SQL queries against dBASE files is relatively slow. SQL on large dBASE files is particularly slow since BDE reads the entire table locally, and then performs the SQL on it. In addition, the BDE SQL simulation onto dBASE is very inefficient. It just can't rival the optimizers of true SQL database.

In contrast, sorting browse columns on large SQL databases is unbelievably fast, as the optimized ranged SQL query is performed very quickly by the server. On a 500,000 record history record, sorting the 2500 history records of Ron Harris by USER takes about 2 seconds on MS SQL Server. Please do not try this on a dBASE file as it will take a very long time, even if you have enough disk space locally. Sorting columns of dBASE files should only be performed on small tables (less than 1MB).

Sorting columns (both on dBASE and SQL) actually creates a dead SQL query, which is a snapshot of how the data looked like when the query was generated (i.e., when the user clicked on the sort column header). Under dBASE, BDE actually creates a temporary dBASE file in the TEMP directory to store the rows of the dead query. If you sort your Pending tab, and someone schedules an activity for you, that new activity will *not* be visible to you until you issue a new query (i.e., sort on another column), or exit and re-enter that contact's Pending tab to restore a live query. Live queries which are used throughout GM4 fetch data in real-time.

Issuing SQL queries from the new 'SQL Query' tab on the Query window against large dBASE files is bound by the same limitations as sorting columns. Simply put, any SQL query against large dBASE files will perform very slowly, even when the SQL query is optimized. Complex JOIN queries on dBASE will perform even slower. Here too, issuing optimized SQL queries against SQL databases will perform very well. A separate document will cover the new SQL Query tab in detail.

And a final note about SQL performance. Don't get your hopes up about stellar performance on SQL databases. Sorting columns is amongst the few areas where GM4 performs faster on SQL databases than on dBASE. Most of GM4's operations are faster on dBASE files, especially when the total GoldMine data occupies less than 1GB of disk space.

DOC6: GM4's Licensing

The GoldMine 4.0 product family offers the utmost in licensing flexibility for both small businesses and large enterprises. GoldMine 4.0 is sold to an organization with one serial number license, the Enterprise License. The organization can then create sub-licenses for each of its remote offices, and for each remote user,

changing the licensing configurations as required over time. The single distributed Enterprise License serves as the authentication mechanism for easy and secure synchronization across the entire organization.

Imagine a company, ABC Corp. with its headquarters in New York, with a networked office of 30 users. ABC also has offices in Boston and Seattle with 10 users each. All 50 people in the organization have their own notebooks and will sync from time to time.

ABC would purchase a 50 user GoldMine 4.0 license and a 50 user GoldSync 4.0 license. These licenses would be installed once, on the headquarters' network, and serve as the seed Enterprise License. Using the License Manager in GoldMine 4.0, ABC will create 2 Site Licenses of 10 users each, one for Seattle and one for Boston. These sub-licenses will be installed on the networks in those cities, and serve as Site Licenses of the original Enterprise License. From the Enterprise License, or any of the Site Licenses, ABC can then create sub-licenses for each of the remote users' notebooks, the Undocked License. All these licenses throughout the organization share the SAME GoldMine serial number.

This flexible Enterprise Licensing hierarchy not only simplify GoldMine's licensing, but it also provides the Enterprise and Site Licenses with the ability to control the security and content of each of their remote Undocked users!

A GoldMine 4.0 license looks like this: C-0050-12345678. The first character denotes the license or sub-license type. The first set of numbers denote the license count, 50 users in this case. The last set of numbers denote the serial number itself, 12345678. The Site license for the Boston office would be S-0010-12345678-BOSTON. The Undocked sub-license for user JON would be U-0001-12345678-JON. The GoldMine 4.0 License Manager shows all the remote sites controlled by the current license. The License Manager of the GoldMine system running in ABC's headquarters in New York will show entries similar to:

```
C - 0050 - 12345678
S - 0010 - 12345678 - BOSTON
S - 0010 - 12345678 - SEATLE
U - 0001 - 12345678 - JON
U - 0001 - 12345678 - JANE
```

The License Manager of the Site License in Boston will show:

```
S - 0010 - 12345678 - BOSTON
U - 0001 - 12345678 - MARK
U - 0001 - 12345678 - MARY
```

The first character of the license denotes the license type. Following are the Enterprise License types available from GoldMine Software. An Enterprise license must only be installed on the organization's primary network.

```
E-xxxx    GoldMine dBASE/SQL/GoldSync, xxxx users and xxxx GoldSync sites.
C-xxxx    GoldMine dBASE/SQL, xxxx users.
D-xxxx    GoldMine dBASE, xxxx users.
G-xxxx    GoldSync, xxxx sites.
```

Following are sub-license types, created by the Enterprise License for remote sites and remote users of the organization:

```
U-0001    GoldMine, Undocked user.
S-xxxx    GoldMine, xxxx users.
Y-xxxx    GoldSync, xxxx sites.
```

The database license type (SQL and/or dBASE) of all sub-licenses is inherited from the Enterprise License. If the Enterprise License is for dBASE only, all sub-licenses will also be dBASE only. The Enterprise License Authentication Seed is also inherited by all sub-licenses to assure that all the organization's sub-licenses are authenticated before synchronization.

Creating a new sub-license is easy. From the License Manager, select either 'New Site' or 'Undocked User'. These option will produces the sub-licenses to use when installing GoldMine on the remote site networks and undocked notebooks. When installing GoldMine on each of the remote computers, enter the appropriate sub-license number produced for each remote site by the Enterprise License. It is important to create and install the sub-licenses correctly, as they control the security of the organization-wide synchronization.

Increasing ("bumping") the Enterprise License is also easy. From the License Manager, select 'New License" and enter a new GoldMine or GoldSync serial number. That's it. Site Licenses and Undocked Licenses cannot be bumped, only the Enterprise License can. Converting a dBASE license to an SQL license requires a GoldMine c/s SQL license with the same license count (or smaller) as the current dBASE license. The current "D" type license will be converted to a "C" license with the original s/n and the c/s license count. Remote sites wishing to take advantage of the new c/s license will have to re-enter their site licensing information. GoldMine Software offers a discount for SQL licenses used to convert existing dBASE licenses.

The license file, LICENSE.DBF, must exist only in the GoldMine "root" directory of each GoldMine installation. If the LICENSE.DBF file is deleted, GoldMine will prompt the user for a license number when they next log in. It is important NOT to delete and re-create the master Enterprise License file, since a new Enterprise License file creates a new authentication seed which will be incompatible with all the existing sub-licenses currently using the authentication seed from the original Enterprise License file.

During the GoldMine 4.0 installation, GoldMine looks for a special self-extracting EXE to run before completing the installation. This self-extracting EXE can be used to install specific pre-prepared data for each remote site, allowing an organization to completely customize the installation for each remote site, while still using the standard GoldMine installation.

GoldMine 4.0 truly sets a new standard in enterprise licensing, synchronization and security. With the new synchronization capabilities of GoldSync 4.0, the Territory Realignment feature to shuffle territories of remotes sales people, and the new flexible security system, distributed organizations gain significant control and flexibility over how their corporate data is replicated and protected.

DOC7: SQL Rehosting 101

This document covers the basic procedures required to re-host GoldMine 4.0 data on SQL client/server databases.

First, a brief overview of GoldMine's directory structures. GoldMine stores its database files in two or more directories. The contact files, with their associated history and profiles reside in their own directory, so that GoldMine can use multiple contact files. The 'GoldMine database' files contain data which is not

contact specific, like the InfoCenter, Forms, Filters, etc., are stored in the GMFiles directory, and identified by the GoldDir= entry in the GM.INI file.

In GoldMine 4.0, there is another directory, named the "root" directory, where the license file, reports and other non-data files are stored. This directory defaults to the "GoldMine directory" when GoldMine is installed. On a network installation, all users must use the same root directory, which must be on the network file server (not the SQL server). A SysDir= entry in the GM.INI file indicates the root directory.

On SQL servers, the data files are not stored in a file directory, but on server databases who's location is unknown to GoldMine. GoldMine identifies these SQL databases using their database alias, as defined using the BDE Configuration utility. The 'GoldMine database' and 'Default contact database' comboboxes on the Preferences Login tab list all database aliases visible to GoldMine.

On SQL databases, GoldMine can use the same SQL database to store both the GoldMine files, and one set of contact files. In fact, for companies using one primary contact database (like GoldMine Software is) this is preferred, as the SQL DBA (database administrator) has only one GoldMine database to maintain, and querying one databases is simpler than joining tables from two databases.

USER NAMES and LOG-ON ACCOUNTS

To log onto an SQL databases, GoldMine has to travel through three layers of user accounts in order to access its tables. These are the SQL log-on user account, the SQL tables owner name, and the GoldMine user name.

GoldMine's default SQL database log-on account is taken from the USER NAME specified in the BDE configuration database alias. MS SQL Server and other servers require that each user account which accesses the GoldMine tables be granted permission. On MS SQL use the Object | Permissions option from the Microsoft SQL Enterprise Manager.

The "table owner" name refers to the user name who created the tables, that is, the SQL user account which was logged on when GoldMine created (or rebuilt!) the databases. In general, all access to tables must be in form <owner>.<table name>, like JON.CONTACT1. If the logged-on user is also the owner, then the <owner> portion need not be specified. However, for users other than the owner, GoldMine must send the queries with owner.tablename aliases. The GoldMine SQL alias is formatted as:

```
DRIVER: BDE_ALIAS: OWNER:   for example,  MSSQL: GM4DB: JON:
```

When new databases are created via Tools | Create Databases, the appropriate alias is saved in the Contact Files profile. On the SQL Query tab, your SQL commands must include the table's owner prefix, for example:

```
SELECT * from JON.CONTACT1
```

For installation where all GoldMine users use the same SQL log-on account in the BDE Configuration (which simplifies this considerably), you can manually drop the OWNER portion from the alias, from the Contact File properties dialog. GoldMine will then log-on all SQL users as the single-account table owner, so SQL commands need not include the owner name. The following would then be a valid SQL:

```
SELECT * from CONTACT1
```

Creating a single SQL log-on account, like GOLDMINE, will considerably simply creating user accounts and granting access on the SQL server, and will not require the OWNER prefixes on SQL commands. It is also recommended that the SQL tables owner be named GOLDMINE, so access to the data from external applications would be more intuitive. A developer can write:

```
SELECT company, contact from GOLDMINE.CONTACT1
```

CREATING SQL DATABASES:

To re-host the GoldMine data onto any SQL server, first create a new database on the SQL server (see details below) and then set up an alias in the BDE Configuration to refer to this new database. To re-host the GoldMine data, select Tools | Create Databases and follow the Wizard.

Creating SQL databases is very different on each vendor's SQL server. Following are brief guidelines to setting up Microsoft SQL Server 6.5, Oracle 7.x, InterBase 4.x and ODBC databases. This section assumes that you have already installed and connected to your SQL server.

MICROSOFT SQL SERVER:

Open up the Microsoft SQL Enterprise Manager and log on. The supervisor username/password is sa/password (you should add yourself as a user via Manage | Logins). To create a database, you must first create a new device at least twice the size of your GoldMine data (highlight the "Database Devices" folder, right click and select New Device...). Then create a new database on this device (highlight the "Databases" folder, right click and select New Database...). Name your new database GM4 or GoldMine, and place it in the newly created device. Go back to Manage | Logins and "Permit" yourself (and others) onto the GoldMine database and the master database. Make sure that the GoldMine database is your default device (for now). Under the "Databases" folder, highlight the GoldMine database, right click and select Edit... Under Permissions make sure you have all permissions. Under options, check 'Select into bulk copy' and 'Truncate Log at checkpoint'. Close down the SQL Enterprise Manager.

To set up an alias for this MS SQL database in BDE (which all applications can use), open the BDE Configuration utility and add a new MS SQL alias. For now, enter only the following properties:

```
Database Name:    GM4   (the database name)
Server Name:      NT4   (the NT server name)
User Name:  JON   (the owner of the database)
TDS Packet size:  4096
```

INTERBASE

Open up the InterBase Server Manager and log on. The supervisor username/password is sysdba/masterkey (you should add yourself as a user via Task|User Security and then logout and log-on as yourself). To create a database, select Task | Interactive SQL, then File | Create Database and enter its filename. To create a local database enter the path like c:\gm4.gdb. To create it on an NT server, enter the UNC path like \\NT4\GM4\gm4.gdb. You now have an InterBase database that YOU own. You can GRANT access to others using InterBase Interactive SQL application (ISQL). Close down the InterBase Server Manager and interactive windows.

To set up an alias for this InterBase database in BDE (which all applications can use), open the BDE Configuration utility and add a new InterBase alias. For now, only enter the following two properties:

```
Server Name:   \\NT4\GM4\GMCF.GDB   (the database path)
User Name:     JON                  (the owner of the database)
```

InterBase does not support the OWNER.TABLE concept. Although the InterBase documentation states that it can be done, we cannot send a SELECT * from JON.CONTACT1 command to InterBase, not even from the InterBase Interactive SQL. The InterBase owner may be at the database level, and not at the table level.

Using the InterBase Server Manager you can create new user accounts. You can have all GoldMine users use one login account, or each user can have their own account. Each workstation can define the default SQL login user in the BDE InterBase alias setting from the BDE Configuration Utility. A default database password can be set in the Preferences Login tab. GoldMine will prompt the user for a the SQL login/password if the defaults do not provide access.

The advantage of using a single SQL login account is that a DBA can easily set the access and privileges to all users at once. Using multiple accounts requires the DBA to GRANT appropriate rights for each user.

ORACLE:
=======
Open up the Oracle7 Navigator and log on. The supervisor username/password is system/manager or po7/po7 (you should add yourself as a user under the Database folder, highlight the User folder, right click and select New...). It's very difficult to create a new database under Oracle, so use the pre-installed database. To set up an alias to the Oracle database, open up the SQL*Net Easy Configuration and select 'Add new alias'. Enter an alias name like ORA_GM4 and select TCP/IP as the protocol. For Local Oracle, your TCP/IP Host Name is 127.0.0.1. Leave the Database Instance as ORCL. Go back to Oracle7 Navigator and create a new database connection under Database Connections. Enter the Oracle alias name you gave above, and enter your username and password. Make sure that you can connect to the database by clicking on it. Close down Oracle7 Navigator.

To set up an alias for the Oracle database in BDE (which all applications can use), open the BDE Configuration utility and add a new ORACLE alias. For now, enter only the following properties:

```
Server Name:      Ora_GM4     (the Oracle database alias)
User Name:  JON         (the owner of the database)
Net Protocol:     TNS
```

ODBC:
=====
Configure your ODBC drivers from the Windows Control Panel ODBC 32-bit appellate.

To set up an alias for ANY ODBC database in BDE (which all applications can use), open the BDE Configuration utility and add a new ODBC alias. For now, enter only the following properties:

```
Database Name:    <blank>           (leave it blank)
User Name:  JON         (the owner of the database)
ODBC DSN:   GM4_MSSQL   (the ODBC Data Source Name)
```

DOC8: The SQL Query Tab

This document covers GoldMine 4.0's new SQL Query tab, which allows users to send native SQL queries directly to the SQL databases, and display the resulting records linked to their appropriate contacts. SQL queries can be extremely useful, not only to query and view related data, but also to dump the contents of any GoldMine table, like a database browser would. The SQL Query tool is one of the most powerful new features in GoldMine 4.0.

SQL (Structured Query Language) is a database language which is understood by all Client/Server SQL databases. While each SQL server uses its own dialect of SQL, they are all compatible with the base SQL-92 standard.

SQL queries are instructions sent to the SQL database to return data. The SQL SELECT command is used to query the data, and although it only employees 10 or so keywords, the SELECT commands can process some extremely complex queries.

GoldMine's SQL Query tool is very powerful, especially on SQL databases. It allows users to enter simple and complex SQL queries, and have the results automatically linked to the appropriate contact records. Users can enter SQL queries on-the-fly, or save queries for later execution. You can send SQL queries against ANY GoldMine table (see examples below).

SQL queries can be executed fast or slow, depending on the composition of the query in relation to the requested data and available indexes. Tuning SQL queries is part art, part science, and different SQL servers might optimize the same SQL query quite differently. There are many developers out there who's primary job function is to develop optimized SQL queries. Your local bookstore has many books covering SQL, one of the best for beginners is the LAN Times "Guide to SQL". The BDE and InterBase documentation also offer help files on SQL commands. BDE also includes some innovative SQL concepts beyond the standard SQL-92, which allow for powerful cross backend queries.

SQL queries on large dBASE tables can be very (very!) slow, since the SQL is actually emulated on a temporary dBASE file which is reconstructed on the local client machine. On the other hand, SQL servers' claim to fame is that they can execute fine tuned queries very fast, even on databases with millions of records.

Following are a few examples of SQL queries, all of which can be executed from GoldMine's SQL Query tool. Examples 2 and 3 demonstrate how to JOIN (relate) the parent contact1 file to its child records in CONTACT2, CAL, CONTHIST and CONTSUPP. Example 4 shows an SQL query on non-contact data.

Example 1: Contacts in L.A.:

```
SELECT * FROM CONTACT1
WHERE Phone1 LIKE "(310)%"
```

Example 2: Contacts with Internet email addresses:

```
SELECT c.contact, s.contsupref, c.phone1, c.city
FROM    contact1 c, contsupp s
WHERE s.rectype = 'P' and s.contact = "Internet Address"
and c.accountno = s.accountno
```

Example 3: Contacts with purchases of over $1,000

```
  SELECT c.company, h.ondate, h.duration, h.ref
  FROM contact1 c, conthist h
  WHERE h.srectype = 'S' and substring(h.duration from 4 for 1) <> ""
  and c.accountno = h.accountno
```

Example 4: All your saved SQL queries

```
  SELECT name, userexp
  FROM filters
  WHERE rectype = 'Q' and userid = "JON"
```

For SQL commands on dBASE files, GoldMine selects one directory to be the default directory for the tables aliased in the SQL. If any of the CONT* files is selected in the FROM clause, that contact files directory is the default, otherwise, the GoldMine directory is the default directory. You cannot have unaliased tables from both directories in the same query, but you can qualify the table names with their dBASE path. The following SQL query against dBASE files in two directories is perfectly legal. It joins the CONTACT1 and CAL tables to list all of JON's activities, shown with the contact name and the state in which they are at:

```
  SELECT  c1.contact, c1.state, ca.ondate, ca.ref
  FROM     'c:\gm4\common\contact1' c1,  'c:\gm4\cal' ca
  WHERE  c1.accountno = ca.accountno AND ca.userid = "JON"
```

DOC9: GM4's dBASE Parser

Over two year ago, when we chose BDE as GoldMine's future database engine, a huge problem prevented us from getting started. BDE does not offer a dBASE parser to evaluate dBASE expressions. But GoldMine's existing data has dBASE expressions in it, in tables like FIELDS, FILTERS, EXPORT and others.

Guy volunteered to take on the arduous task of writing a full-blown fully optimized dBASE parser. Skepticism disappeared a few months later when his parser plugged right in without a hitch. Since then, we have been optimizing the new GoldMine 4.0 dBASE parser, and we have some good news for you.

A GoldMine 4.0 dBASE expression is no longer limited to 256 bytes. The expression is no longer limited to 32 components or levels, it can be as complex as you wish. All CodeBase expression functions are supported.

GoldMine 4.0 now includes a dBASE Expression Tester for you to test dBASE expressions. To activate the tester, press Ctrl-Shift-D when viewing the Record window.

Filter expressions function equally well on dBASE or SQL tables. On SQL, the dBASE filter is evaluated on the client side, not the server side. Filters which access only CONTACT1 fields have been optimized to run up to 5 times faster on GoldMine 4.0 as compared to GoldMine 3.2.

The GoldMine 4.0 dBASE parser also offers some welcomed new functions, including AT, RAT, PADR, WDATE and a few others. There is also support for a RANDOM and SEQUENCE expressions.

Following is a summary of the new functions:

ALLTRIM(<string>)
trims spaces from the right and left ends of the string.

AT(<string1>,<string2>)
returns the first position of <string1> in <string2>.

COUNTER(<name>, <inc>, <start>, <action>)
COUNTER works similarly to the SEQUENCE function. The key difference is that COUNTER stores the count value between GoldMine sessions, and is shared by all GoldMine users. The SEQUENCE counter is local to the operation, and its count is lost at the end of the operation. Since the COUNTER function updates a database counter, it is much slower than SEQUENCE which updates a memory counter.

COUNTER returns a sequence of consecutive numbers each time the expression is evaluated. The counter <name> must be unique and up to 10 character long. Each evaluation of the function increments the counter by the <inc> value. The <start> and <action> parameters are optional. When <action> is 1, the <start> value is used to reset the counter. The counter is deleted when <action> is 2.

GoldMine can track and unlimited number of counters, each with a unique name. The counter values are stored in the LOOKUP table. Counters can be used anywhere dBASE expressions are allowed, including F2 lookups. The following F2 entries can be placed in the Incoming Call Reference F2 to number each type of call.

```
~"T"+str(counter("ITech",1))
~"S"+str(counter("ISale",1))
```

LEN(<string>) returns the length of a string
LENGTH(<string>) same as LEN()

PAD(<string>, <length>, <mode>)
PAD(<string>, <length>, <fill>, <mode>)
PADL(<string>, <length>, <fill>)
PADR(<string>, <length>, <fill>)
returns <string> padded to <length> with the <fill> character. <fill> is optional and defaults to a space. PADR pads the string to the right while PADL pads from the left. For PAD, <mode> can be 0 for right pad (the default), 1 for centered and 2 for left pad.

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. For example, RANDOM(10) returns a number between 0 and 9. The <range> parameter defaults to 32,761 if not specified. Random numbers up to 2 billion can be achieved using the expression random(32761) * random(32761).

RAT(<string>,string>)
returns the last position of <string1> in <string2>.

SEQUENCE(<start>, <inc>)

returns a sequence of consecutive numbers each time the expression is evaluated. When the expression is first evaluated, the <start> parameter initializes the counter. Each subsequent evaluation of the function increments the counter by the <inc> value.

To generate numbers in a specific range (100-200, 3-9, etc...) the following expression can be used:
sequenceStart + SEQUENCE(0,1) % sequenceLength. The sequence length should include the actual number of numbers in the range. for example:

```
100-200: 100 + SEQUENCE(0,1)%101
3 - 9:      3 + SEQUENCE(0,1)%7
```

SEQUENCE can be called anywhere an expression can be entered. For example, Global Replace can be used to number filtered contact records. The SEQUENCE(1,1) expression in a Field View shows a real-time indication of how many times the Record window was updated. The sequence count is local to each operation. In a global replace, the count is initialized on the first record and killed on the last. On the Record window (as a field) the count is initialized when the window is created, incremented for every paint, for the life of the window.

STR(<value>,<length>,<fill char>)
STR(<value>,<length>,<decimals>,<fill char>)
returns the numeric <value> formatted as a string. All parameters except <value> are optional. The <length> parameter pad the number to the left with spaces, or the with <fill char> if specified.

WDATE(<date>, <format>)
returns the <date> formatted in variety of ways, based on the optional parameter <format>.

<format>		
0	mmm dd, yy	Jan 22, 97
1	ddd, mmm dd, yy	Thu, Jan 22, 97
2	mmm dd	Jan 22
3	Long date style	Thursday, Jan 22, 1997

The Long date style format 3 is taken from the Windows Regional Settings.

Examples:

```
WDATE(  DATE(),  3 )    => Thursday, Jan 22, 1997
WDATE(  Cal->OnDate )   => May 3, 96
```

DOC10: GM4's DataStream

The GoldMine 4.0 DDE server has gained a very powerful new DDE command, DataStream.

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. An optional filter can also be applied to the data set.

The DataStream method allows for many useful applications. One such group of applications which will probably appear soon are applications which publish the

contents of GoldMine data on the Internet by merging HTML templates with the data returned by GoldMine's 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, from the addresses of its dealers in a particular city, to financial numbers stored in Contact2, to the 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 4 sub-commands, each taking different parameters. Following are the sub-commands, in the order in which they must be called:

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

The "range" OR "query" sub-commands must be called first to request the data. The "range" and "query" sub-commands return an integer handle, iHandle, which must be passed to the "fetch" and "close" sub-commands. You must use "range" OR "query", not both.

[DataStream("range", sTable, sTag, sTopLimit, sBotLimit, sFields, sFilter, sFDlm, sRDlm)]
The "range" sub-command returns a range of records based on an index. 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 (refer to a section below detailing the sFields string format). The other "range" parameters are optional.

[DataStream("query", sSQL, sFDlm, sRDlm)]
The "query" sub-command sends the sSQL query for evaluation on the server. The SQL query can join multiple tables and return any number of fields. The optional sFilter parameter can specify a boolean dBASE filter expression to apply to the data set (even on SQL tables), similar to the DDE SETFILTER command. The optional sFSep and sRSep parameters can override the return packet's default field and record delimiters of CR and LF.

[DataStream("fetch", nRecords, iHandle)]
The "fetch" command returns a single packet string containing 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. Please refer to the section below detailing the packet format.

[DataStream("close", iHandle)]
The "close" command 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!).

FIELD SELECTION
===============
The sField parameter passed to the "range" sub-command should consist of the field names and dBASE expressions to evaluate against each record in the data set. Each

field must be terminated with the semicolon ; character. dBASE expressions must be prefixed with the ampersand & character and terminated with a semicolon. For example, the following commands requests 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)]
```

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 Angles|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 in size. It takes DataStream about as much time to read 3 records as it does to read 30. For best performance, adjust the number to records requested by the "fetch" command to return 8K to 32K packets.

PERFORMANCE
==========
DataStream is the absolute fastest way to read data from GoldMine tables! Used correctly, GoldMine's DataStream will return the data faster than most development environments would directly. Consider for the following advantages DataStream offers:

1. DataStream issues a single, most efficient SQL query or dBASE 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, greatly minimizing network traffic.

2. All fields and expressions are parsed initially by the "range" and "query" commands, and 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 its data is read. This makes a big difference when reading hundreds or thousands of records.

3. Only 3 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 C++, the fastest way to fly. 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 must faster on dBASE tables than the "query" command does.

The following DataStream command returns all e-mail addresses in the current contact file. On my P133, DDEReq32 returns 312 records in under 1 second flat!

```
 [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 Software, add a filter to the "range" command:

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

The following DataStream returns all entries from all F2 lookups. The fields are delimited with a comma, and the records with the default LF. My P133 picks up all 847 lookup records (in a single 23K packet) from an MS SQL table in just 1.8 seconds. Dave's server will probably do a lot better on a good day.

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

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", "","",")]
```

DOC11: GM4's Word Links

GoldMine 4.0's Word links are now installed on-the-fly, when the user merges their first form. GoldMine 4.0 installs with about 20 template merge forms, which can be customized by each user.

GM4Setup.exe installs all standard merge forms to the \GoldMine\Templates directory. When the user opens the Merge Forms window for the first time, they can launch any of the public merge documents available.

The Microsoft document templates are installed for user public, without an AppID or DDE command. When the user launches a .DOC or a .DOT (or an .XLS), GoldMine looks for the word.document.8 entry in the Registry, and then for the word.document.6 entry if the first entry is not found. GoldMine then verifies that the Word link is installed by looking for a 4.6 or 4.8 entry under the Software\Microsoft\Word\ GMLinkVersion section of the Registry. If it's not there, GoldMine will ask the user if it's ok to install the Word link.

When the user tries to modify a default template, GoldMine copies that template to the user's template directory (i.e., \GoldMine\Templates\UserName) and creates a new record in the FORMS table under the current user name, with the default template's properties. The user now has a copy of the original template for them to customize.

To simplify specifying directories, GoldMine looks for the template filename under the \GoldMine\Templates directories, so the complete path of the templates need not be specified. The public templates have no path since they are stored under the \ Templates directory, and user's customized templates will default to UserName\ docname. GoldMine first checks for the document in its absolute path if a drive letter or UNC is provided, then under the \Templates directory, then under \ Templates\UserName, and finally under the GoldMine root directory.