



# COLLECTION MASTER

SQL-SYNC 2.1A.002  
is Here!

Presented by  
Luis Gomez



**MASTERMIND**  
**SERIES**





Presented by

**VERTICAN**  
TECHNOLOGIES



# Resources

-  **MASTERMIND**  
SERIES Sessions and Collection-Master How To docs:
  - [How to Create a CM EDI RT171 Using SQL Sync](#)
  - [How to Plan for Data Recovery](#)
  - [SQL-SYNC - Presentation – Video](#)
- Current  Articles:
  - <https://www.vertican.com/vconnect/#tips>
- Help Center
  - <https://vportal.vertican.com/Help-Center.aspx>



# SQL-SYNC – 2.1A.002

- New Features CM 9.1L.008
  - Brand new manual replaces the existing documentation.
  - Support for modern ODBC Drivers, including TLS encryption
  - Sync\_Timer updated to provide more information
  - New fields Date\_est & Date\_upd added to each table
  - Schedule now supports 0 minutes (10 seconds)
  - Tracking implemented when Deleting Records
    - Particularly helpful for tracking Purging and Closing / Re-Opening claims
  - Changes Enhanced to include NEW\_VALUE
  - Apply Uncommitted Changes and Deletes
    - The CHANGES table as well as \*\_DEL tables will update SQL-SYNC providing INCR\_SYN: functionality as often as every 10 seconds.





# Business Rules 4.31hdd

- Business Rules is the language that runs Collection-Master
  - Microsoft Visual Studio 2022
  - Microsoft C++
  - Recently compiled with updated CURL & OPENSSL libraries
  - Client Server now leverages TLS 1.2
  - HTTPS connections use the new libraries, and support TLS 1.2
  - SQL-Server enhanced to support newest ODBC Drivers.
  - BR 4.31hdd compiled 2/5/2024





# Support for Modern ODBC Drivers

- Prior versions of SQL-SYNC only supported the **“SQL Server”** drivers
- Vertican collaborated with Business Rules! (4.32hdd) to refresh the libraries
  - OpenSSL library now supports TLS 1.2
  - SQL Server Drivers enhanced to support modern ODBC Drivers
    - Legacy Drivers
      - SQL Server
      - SQL Server Native Client 10.0 – This Driver is typically installed with vMedia
      - SQL Server Native Client 11.0
      - ODBC Driver 11 for SQL Server
      - ODBC Driver 13 / 13.1 for SQL Server
    - Modern Drivers – (Recommended)
      - ODBC Driver 17 for SQL Server
      - ODBC Driver 18 for SQL Server
  - Modern SQL Server Drivers support Encryption with SQL Server
  - Make sure you are running SQL-Server 2016 or 2019
- Remember that SQL-SYNC runs on the Collection-Master Server as a 64-bit Application.



# Support for modern ODBC Drivers



- SQL Server Native Client 10.0
  - This driver was often provided with “Other Installers”
  - There were several versions, make sure you have the most recent
    - Version = 2009.100.6560.00
    - Date = 12/28/2017
- ODBC Driver 18 for SQL Server – (Recommended)
  - Latest version as of 2024
  - Supports SQL Server 2008 to SQL Server 2022
  - Added support for new TLS protocols, improved error messages, security features, and other performance improvements



# MapDrive.\_cs

```
Rem [---- Start of SQL-SYNC-SETUP ----]

SQL-Sync-SERVER = Server Name (E.g. CLS-SQL)
SQL-Sync-DATABASE = Database Name (E.g. CM)

Rem [---- Optional Parameters ----]

SETENV SQL-SYNC-DRIVER {ODBC Driver 18 for SQL Server}
SETENV SQL-SYNC-ENCRYPT YES
```

- Edit F:\CLSINC\WBWIN\Mapdrive.\_CS
- SQL-SYNC-DRIVER
  - Specify the Desired Driver
- SQL-SYNC-ENCRYPT – (Not Available for the **SQL Server Driver**)
  - YES is the only option for this value (Require an encrypted connection)





# Sync\_Timer Updated

SQL-SYNC- 2.0A003 - Session ID:1 - - Session 721

Session ID: 1

Bulk\_Insert.wb

Program: Sql-Sync\Bulk\_Insert.wb INCONVENIENT

Run Time: 2024/05/30 09:15:25

Current Time: 2024/05/30 09:15:18

ID	Event	Last Run	Next Run
124	BULK_INS:INCONVENIENT (Closed - Legal)	2024/05/30 09:10:25	2024/05/30 09:15:25
130	BULK_INS:LEGPROC (Closed - Legal)	2024/05/30 09:10:26	2024/05/30 09:15:26
160	BULK_INS:ACTIVE_DEL,MASTER_DEL,DEBTOR_DEL,DPHONE_DEL,INFINITY_DEL,DCHANGES_DE	2024/05/30 09:10:28	2024/05/30 09:15:28
115	BULK_INS:DSERVE,SERVEACT,FILESERVE (Closed - Legal)	2024/05/30 09:10:32	2024/05/30 09:15:32
141	BULK_INS:ADDLNOTES_PF (Open - Legal)	2024/05/30 09:10:40	2024/05/30 09:15:40
23	BULK_INS:ACT_ARCH,DCHANGES,CHANGES (Closed - Closed Slow Tables II)	2024/05/30 09:14:44	2024/05/30 09:15:44
146	BULK_INS:ADDLNOTES_PF (Closed - Legal)	2024/05/30 09:10:45	2024/05/30 09:15:45
13	BULK_INS:ACT_ARCH,DCHANGES,CHANGES,DBILLING (Open - Slow Tables II)	2024/05/30 09:14:46	2024/05/30 09:15:46
156	BULK_INS:ACTIVE_DEL,MASTER_DEL,DEBTOR_DEL,DPHONE_DEL,INFINITY_DEL,DCHANGES_DE	2024/05/30 09:14:49	2024/05/30 09:15:49

RUN | SINC\SQL-SYNC\SYNC\_LIBRARY\* | 50560:01 | NC\_TIMER | 12906 | 4.31h





# Sync\_Timer Updated

- UI facelift
- New GRID showing the next nine scheduled events
- Enhanced Error trapping:
  - Will retry or restart when SQL-Server Times out.
    - These errors are SQL Server failures, often due to resource issues.
    - The automatic retry will allow the SQL Server to continue if the problem is temporary.

SQL-SYNC: 204009 - Session 011 - Session 721

Session ID: 1

Bulk\_Insert.vwb

Program[Sql-Sync]Bulk\_Insert.vwb

Run Time[2024/05/30 09:15:25]

Current Time[2024/05/30 09:15:18]

INCONVENIENT

ID	Event	Last Run	Next Run
124	BULK_INS:INCONVENIENT (Closed - Legal)	2024/05/30 09:10:25	2024/05/30 09:15:25
130	BULK_INS:LEGPROC (Closed - Legal)	2024/05/30 09:10:26	2024/05/30 09:15:26
160	BULK_INS:ACTIVE_DEL:MASTER_DEL:DEBTOR_DEL:PHONE_DEL:INFINITY_DEL:DCHANGES_DE	2024/05/30 09:10:28	2024/05/30 09:15:28
115	BULK_INS:OSERVE:SERVE:ACT:FILESERVE (Closed - Legal)	2024/05/30 09:10:32	2024/05/30 09:15:32
141	BULK_INS:ADD:NOTES_Pf (Open - Legal)	2024/05/30 09:10:40	2024/05/30 09:15:40
23	BULK_INS:ACT_ARCH:DCHANGES:CHANGES (Closed - Closed Slow Tables II)	2024/05/30 09:14:44	2024/05/30 09:15:44
146	BULK_INS:ADD:NOTES_Pf (Closed - Legal)	2024/05/30 09:10:45	2024/05/30 09:15:45
13	BULK_INS:ACT_ARCH:DCHANGES:CHANGES:DELLING (Open - Slow Tables II)	2024/05/30 09:14:46	2024/05/30 09:15:46
156	BULK_INS:ACTIVE_DEL:MASTER_DEL:DEBTOR_DEL:PHONE_DEL:INFINITY_DEL:DCHANGES_DE	2024/05/30 09:14:49	2024/05/30 09:15:49

SQL-SYNC: 204009 - Session 011 - Session 721

# Date\_est & Date\_upd Added to Each Table



Field	Description
Date_est	<ul style="list-style-type: none"><li>• Date the record was initially inserted into the table.</li><li>• After installing the SQL-SYNC update, records will have a null value for Date_est as they were added before the field existed.</li><li>• Some users may choose to update DATE_EST with a “Known Date” instead of Null.</li></ul>
Date_upd	<ul style="list-style-type: none"><li>• Date the record was last updated.</li><li>• After installing the SQL-SYNC update, this field will be unknown and populated with Null.</li><li>• As the records are updated, the Date_upd field will also be updated. That means some records will have a Date_upd value while the Date_est remains null.</li><li>• Some users may choose to update DATE_UPD with a “Known Date” instead of Null.</li></ul>



# Date\_est & Date\_upd Added to Each Table

- Date\_est & Date\_upd reflect the date and time SQL-SYNC was updated.
- While it provides insight into when the Collection-Master records were updated, there is no direct correlation.
- Upon installing the new SQL-SYNC version, these fields will be NULL.
- **EXEC** [dbo].[Stp\_Initialize\_Date\_EST\_Date\_UPD]
  - Stored Procedure included, available for your DBA to use to initialize DATE\_EST & DATE\_UPD



# Schedule Now Supports 0 Minutes (10 Seconds)

- There are certain tables that you might want to refresh aggressively.
- Each event is assigned a refresh period in minutes.
- Now, 0 Minutes = 10 Seconds.
  - This means that the table will refresh every 10 seconds rather than refresh “Instantly” with no wait.
  - Remember that the task may take longer than 10 seconds, and in that case, the loop will start immediately upon completing the last event.
- BULK\_INS: is a likely candidate for 0 Minutes:
  - MASTER, DEBTOR, ACTIVE, CHANGES, ACTIVE\_DEL (etc.)
- Use Session 3 (or another session) to run the aggressive BULK\_INS for Uncommitted Changes & Deletes.





# Tracking implemented when Deleting Records

- Collection-Master & SQL-SYNC
- New tables to track details about a deleted record
  - Naming Scheme is the Source Table and \_DEL.
  - Example ACTIVE\_DEL
- Entries in the Delete tables are Added (Inserted)
  - Immediately as records are deleted.
- SQL-SYNC will leverage a BULK\_INS event
  - Perform BULK\_INS events frequently, as often as 0 minutes (10 seconds)



# Tracking Implemented when Deleting Records

- Tables will have their own \*\_DEL table to track deletes.
  - These tables will have an entry in Data (1) and History (2)
- Delete Type provides insight into why the record was deleted.
  - Open/Close
- Some useful \*\_DEL tables to add:
  - ACTIVE\_DEL, MASTER\_DEL, DEBTOR\_DEL, DPHONE\_DEL, INFINITY\_DE L, DCHANGES\_DEL



ACTIVE\_DEL - Notepad

DELETE_TYPE	(O)pen (C)lose (D)elete (P)urge	1	1.00	C	001	
SYS_DATE	Date of Transaction (System)	2	4.00	BH	002	DATE(CCYYMMDD)
SYS_TIME	100ths of Seconds Since Midnight-SYS	6	4.00	BH	003	
DEL_USER_ID	User ID	10	2.00	BH	004	
REC_NUMBER	Record Number Actual Record #	12	4.00	BH	005	
FILENO	File Number	16	8.00	C	006	
TRANS_DATE	Date of Transaction	24	4.00	BH	007	DATE(CCYYMMDD)
TRANS_TIME	100ths of Seconds Since Midnight	28	4.00	BH	008	
CODE	8 Digit Code	32	8.00	C	009	
USER_ID	User ID #	40	2.00	BH	010	
RECNO	Record Number (KEY)	42	4.00	BH	011	



# Changes Enhanced to Include NEW\_VALUE

- In Collection-Master, the CHANGES table has a new field:
  - NEW\_VALUE
  - In Collection-Master, the Changes Log “Calculates this Field”.
  - In SQL-Server, while possible to calculate, it’s challenging.
  - The new field allows users to query the old new values easily.
  - Collection-Master will also leverage this new field.

Field	Description
VALUE	The value before it was changed.
NEW_VALUE	The new value that was assigned to the field.





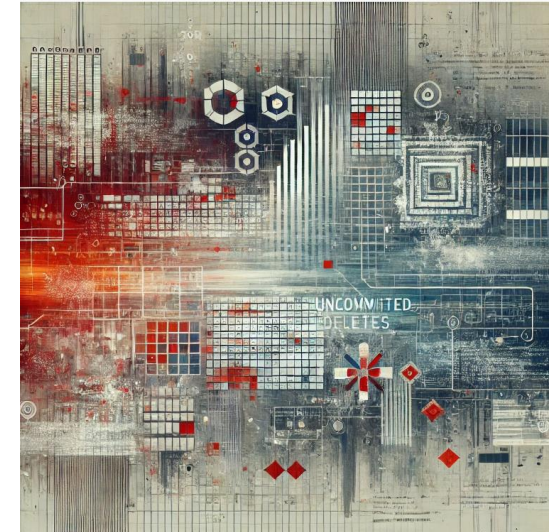
# Apply Uncommitted Changes

- New SQL Server stored procedures:
  - `stp_Apply_Uncommitted_Changes`
    - Executes any time CHANGES inserts new records
    - Set frequency for CHANGES to the desired frequency 1-5 or 0 minutes.
    - Stored procedure will take the newly added CHANGES records and update the corresponding table. With the NEW\_VALUE from CHANGES.
    - Date\_UPD in the destination table will be updated to match the date of the CHANGES record.
    - INCR\_SYN is still required and will ensure the tables are “Fully Synchronized”.



# Apply Uncommitted Deletes

- New SQL Server stored procedures:
  - `stp_Apply_Uncommitted_Deletes`
    - Executes any time a \*\_DEL inserts new records.  
Example: `ACTIVE_DEL`
    - Set Frequency for \*\_DEL to 1-5 or 0 minutes.
    - Stored procedure will take the newly added \*\_DEL records and delete the records from the corresponding table.
    - `Date_upd` is not relevant for Uncommitted Deletes
    - `INCR_SYN` is still required and will ensure the tables are “Fully Synchronized.”





# Updating SQL-SYNC

- When running SQL-SYNC, it will automatically check the version.
  - As appropriate, the table schema is updated.
  - DATE\_EST & DATE\_UPD will be added to each tables.
    - Note: In SQL, these new fields will appear at the end.
  - The Stored Procedures are updated.
  - The update may take some time, depending on how many changes apply.
- Chicken or The EGG
  - SQL-SYNC will update the schema and stored procedures.
  - The Changes table is required, and the setup\_db routine will create the file as appropriate.
  - It's possible that when updating very old systems, you will need to run Setup\_db/SQL-SYNC.
- Stored Procedures are delivered to help manage the database.
  - STP\_Create\_Date\_UPD\_Indexes
  - Stp\_Initialize\_Date\_EST\_Date\_UPD



# RUN Setup\_DB.wb/SQL-SYNC

- RUN Setup\_DB.wb/SQL-SYNC
  - Will set up or update SQL-SYNC to the current version.
  - Routine has been enhanced to provide troubleshooting information when setup fails.
  - Setup will read the MAPDRIVE.\_CS to pick the proper SQL Driver.
- Automatic Update
  - SQL-SYNC will notice when Collection-Master is running a different version from SQL-Server.
  - In these cases, the software will update the SQL Server with the latest version.



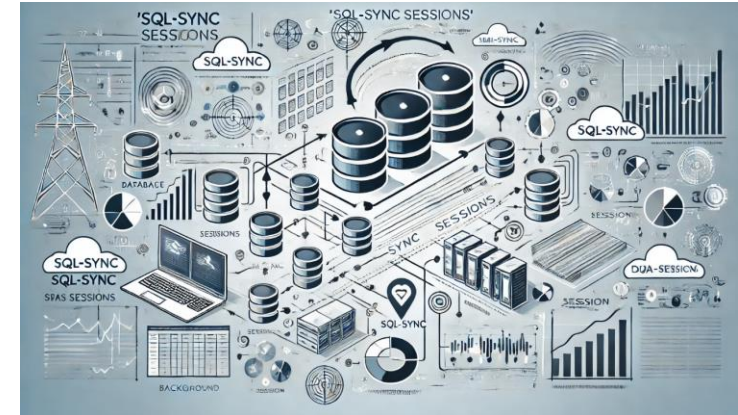
# NextGen Product – Path (Demographics)

- User updates a field in Collection-Master.
  - Update Debtor.Name
- Collection-Master updates the table and creates a record in CHANGES.
- SQL-SYNC performs BULK\_INS: on CHANGES.
  - Changes include the change within 10 seconds.
  - stp\_Apply\_Uncommitted\_Changes is triggered for DEBTOR.
  - Leveraging the information from CHANGES, the dbo.DEBTOR table is updated in SQL with the new name updating DATE\_upd.



# SQL Sync Sessions

- [dbo].[SyncSchedule].[Session\_ID]
  - Supports Values 1-999
  - One Workstation may run up to 9 sessions.
- Typical Session ID Purposes
  - 1: Use for Bulk\_Ins: Typical frequency 5 minutes
  - 2: Use for Inc\_Syn: Typical frequency 60 minutes
  - 3: Use for Bulk\_Ins: Typical frequency 0 or 1-5 Minutes
    - Tables like MASTER, CHANGES, DEBTOR, DIARYINT, ACTIVE\_DEL
  - 4: Use for Inc\_Syn: Typical frequency 5-15 minutes
    - Tables like MASTER, DEBTOR, DIARYINT





# AUTOLOG – Added New Fields

- AutoLog is a table that tracks AUTOMATED tasks.
  - We added new fields to help track user(s)/computer(s) & logins.
  - AUTOMATE tends to run unattended, and the log helps troubleshoot licensing issues.



SESSION	Session ID	108	4.00	C
COMPUTER_NAME	Computer Name	112	20.00	C
TOTAL_USERS	Total Users in Collection-Master	132	4.00	N
CURRENT_LOGIN	Total Concurrent Login for Self	136	4.00	N
LOGIN_NAME	OS Login name	140	32.00	C
AUTOMATE_INI	Name of the Automate.ini script	172	64.00	C





# USERLIST – Added New Fields

- New Fields added to USERLIST to help troubleshoot licensing issues.



VLM_APP_PATH	Path to VLM Application		544		128.00		C	...
VLM_DATA_PATH	Path to VLM Data		672		128.00		C	...
VNUMBER	Client vNumber		800		10.00		C	...
COMPANY_NAME	Client Name		810		80.00		C	...
COMPUTER_NAME	Computer Name		890		20.00		C	...
VLM_EXP_DATE	Subscription Expiration Date		910		10.00		C	...
LOGIN_DATE	Last Login Date		920		10.00		C	...
MAX_TODAY	Concurrent Logins Today		930		4.00		N	...
MAX_SUB	Concurrent Logins Subription Period		934		4.00		N	...
MAX_TOTAL	Concurrent Logins Maximum		938		4.00		N	...



# [dbo].[ufnCateg2]

- Scalar function returns the description for accounting codes.
- Function has been enhanced with “Latest accounting Codes”.

```
WHEN @Code='90.09171' THEN 'CMEDI - Interest Adjustment (09171)'  
WHEN @Code='90.00906' THEN 'iFix - Interest Adjustment (00906)'  
WHEN @Code='90.09899' THEN 'EDI Time Stamp (09899)'  
WHEN @Code='90.09901' THEN 'GE-RMS Time Stamp (09901)'  
WHEN @Code='90.09900' THEN 'EDI Updated Account Card (09900)'  
WHEN @Code='90.09991' THEN 'Stat Fee Stamp (09991)' -- CalcSAtty Uses This  
WHEN @Code='90.09951' THEN 'EDI Updated Suit Info (09951)'  
WHEN @Code='90.09992' THEN 'EDI Updated Judgment Info (09992)'  
WHEN @Code='90.09993' THEN 'EDI Time Stamp (09993)'  
WHEN @Code='90.09999' THEN 'EDI Time Stamp (09999)'  
WHEN @Code='92.00999' THEN 'EDI Time Stamp (00999)'  
WHEN @Code='99.09998' THEN 'Trak BOA No Int (09998)'  
WHEN @Code='99.09997' THEN 'NCODE Interest Stamp (09997)'  
WHEN @Code='90.09996' THEN 'Citibank Financial Interest Stamp (09996)'  
WHEN @Code='90.09995' THEN 'Latitude EAF Interest Stamp (09995)'  
WHEN @Code='80.00001' THEN 'Cost Invoice Generated (00001)'  
WHEN @Code='80.00002' THEN 'Cost Invoice via Gross Remittance (00002)'  
WHEN @Code='90.09994' THEN 'TRAK2 Interest Stamp (09994)'  
WHEN @Code='90.09898' THEN 'TRAK2 Fee Stamp (09898)'  
WHEN @Code='90.08101' THEN 'Payment Schedule Added (08101)'  
WHEN @Code='90.08111' THEN 'Payment Schedle Removed (08111)'  
WHEN @Code='90.10001' THEN 'Client SoR Balance Stamp (10000)'  
WHEN @Code='90.09200' THEN 'Recalculate Interest Stamp (09200)'  
WHEN @SubCode='001' And ( ( @MainCode>=33 And @MainCode<=81 ) Or ( @MainCode>=501 And @MainCoc  
WHEN @SubCode='002' And ( ( @MainCode>=33 And @MainCode<=81 ) Or ( @MainCode>=501 And @MainCoc
```



# [dbo].[ufnCollord]

- Scalar function
  - @Collordi as varchar(18) – Collection-Order from Forwarder File
  - @O\_Collord as varchar(18) – Collection-Order from 1-S-4 Setup
- Returns the appropriate Collection-Order

```
IF CHARINDEX('S',@Collord_Tmp)=0 and CHARINDEX('L',@Collord_Tmp)=0 and CHARINDEX('P',@Collord_Tmp)>0
    SET @Collord_Tmp=REPLACE(@Collord_Tmp,'P','LP')
IF CHARINDEX('O',@Collord_Tmp)=0 and CHARINDEX('P',@Collord_Tmp)>0
    SET @Collord_Tmp=REPLACE(@Collord_Tmp,'P','OP')
IF CHARINDEX('V',@Collord_Tmp)=0 and CHARINDEX('P',@Collord_Tmp)>0
    SET @Collord_Tmp=REPLACE(@Collord_Tmp,'P','PV')
IF CHARINDEX('R',@Collord_Tmp)>0 and CHARINDEX('C',@Collord_Tmp)=0
    SET @Collord_Tmp=REPLACE(@Collord_Tmp,'R','RC')
IF CHARINDEX('N',@Collord_Tmp)>0 and CHARINDEX('C',@Collord_Tmp)=0
    SET @Collord_Tmp=REPLACE(@Collord_Tmp,'N','NC')
IF CHARINDEX('J',@Collord_Tmp)>0 and CHARINDEX('C',@Collord_Tmp)=0
    SET @Collord_Tmp=REPLACE(@Collord_Tmp,'J','JC')
```



# [dbo].[ufnParseAddress]

- @String **NVARCHAR**(64) – Street Address to be parsed
- @Get **NVARCHAR**(64) - Component to return
  - **'City'**
  - **'State'**
  - **'Zip'**
  - **'Country'**
- **Select** [dbo].[ufnParseAddress]('Fairfield, NJ 07054','City') **as** City  
    ,[dbo].[ufnParseAddress]('Fairfield, NJ 07054','State') **as** State  
    ,[dbo].[ufnParseAddress]('Fairfield, NJ 07054','Zip') **as** Zip  
    ,[dbo].[ufnParseAddress]('Fairfield, NJ 07054','Country') **as** Country



# [dbo].[ufnParseName]

- @NameString `varchar(100)`  
--Full Name when '/' is  
Includes assumes Last/First

## Example:

```
Select [dbo].[ufnParseName]  
( 'Doe/John', 'f M L' ) As  
Parsed_Name
```



@NameFormat `varchar(20)`

- P = Full prefix
- p = Abbreviated prefix
- F = First name
- f = First initial
- M = Middle name
- m = Middle initial
- L = Last name
- l = Last initial
- S = Full suffix
- s = Abbreviated suffix
- . = Period
- , = Comma
- [ ] = Space



# [dbo].[View\_Paperless]

- View that combines the various tables and creates readable output.
  - Order by date\_time, recno to match the paperless file order.



Folder_Number	Record_Number	FILENO	DATE_TIME	INIT	CODE	NOTE	BILLED_DATE	RECNO	RECEIVED	DISBURSED
<a href="#">Click to select all grid cells</a>	10318	SAMPLE1	2013-09-03 19:00:53.000	LIG	90.00051	Suit Interest Stamp (00051) Complaint	NULL	2776086	0.00	0.00
1	2310319	SAMPLE1	2013-09-04 09:28:28.000	LIG	RTF-D...	Instant Letter	NULL	154235	0.00	0.00
1	2310320	SAMPLE1	2013-09-04 09:30:05.000	LIG	*ADD-10	Queue search	NULL	0	0.00	0.00
1	2310321	SAMPLE1	2013-09-04 09:30:06.000	LIG	*CLAIM	Claim:9 Fields Modified	NULL	2777702	0.00	0.00
1	2310322	SAMPLE1	2013-09-13 15:16:26.000	LIG	90.00051	Suit Interest Stamp (00051) Complaint	NULL	2188173	0.00	0.00
1	2310323	SAMPLE1	2013-09-13 15:47:39.000	JAC	35	FIRM CHARGE 50.00	NULL	2188195	0.00	50.00
1	2310324	SAMPLE1	2013-09-13 15:47:39.000	JAC	95	Invoice Billed XCOSTINV 50.00	NULL	2188196	0.00	0.00
1	2310325	SAMPLE1	2013-09-17 15:39:02.000	LIG	NJSUIT	testing NJ Suit program	NULL	2193317	0.00	0.00
1	2310326	SAMPLE1	2013-09-17 15:40:29.000	LIG	NJSUIT	testing NJ Suit program	NULL	2193370	0.00	0.00
1	2310328	SAMPLE1	2013-09-21 12:04:44.000	JAC	2.0001	Fees on Direct Payment (1) Cash or Cash Equi...	NULL	2198913	0.00	0.00
1	2310330	SAMPLE1	2013-09-21 12:16:07.000	JAC	2.0001	Fees on Direct Payment (1) Cash or Cash Equi...	NULL	2198915	0.00	55.55
1	2310336	SAMPLE1	2013-09-30 08:27:19.000	GK	XFILED	THIS IS A X CODE	NULL	0	0.00	0.00
1	2310331	SAMPLE1	2013-09-30 12:22:26.000	JAC	95.001	Invoice Billed- (001) -10.00	NULL	2969326	0.00	-10.00
1	2310332	SAMPLE1	2013-09-30 12:22:26.000	JAC	95.002	Invoice Billed- (002) 10.00	NULL	2969327	0.00	10.00
1	2310334	SAMPLE1	2013-10-01 08:06:31.000	GK	*Emplyr2	Added new Employer 2 #: #5	NULL	0	0.00	0.00
1	2310335	SAMPLE1	2013-10-01 08:06:31.000	GK	*Emplyr3	Added new Employer 3 #: #6	NULL	0	0.00	0.00



# RUN SYNC\_FOLDER/SQL-SYNC

- The program will go through each available Table, and perform the following:
  - Make sure the table is created in the SQL-SYNC Database.
  - Review the fields and data types, ensuring that any new fields are added to the table.
    - Note: New fields are added to the end of the table.
- Note: This routine is generally fast, taking only a few minutes, but if there is a schema change that requires SQL Server to update a data type on every record, it might take a long time.
- SYNC\_FOLDER:ALL
  - Add this EVENT in [dbo].[SyncSchedule] to have the schema automatically updated every day

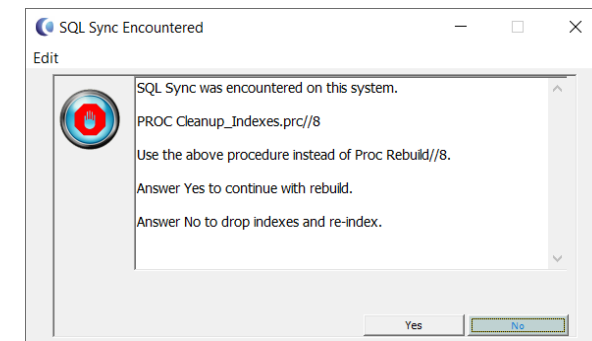
EventId	Event	Folder_Code	Group_Code	TimeOfDay	FreqInMinutes	Start_Record	End_Record	Start_TimeOfDay	End_TimeOfDay	Enabled	Session_ID
58	SYNC_FOLDER:ALL	SQL	SQL	22:00:00.0000000	0	NULL	NULL	NULL	NULL	1	1



# Packing the Database: Optimizing Collection-Master System



- The traditional way to optimize and repair the Collection-Master system involves running a process called PROC REBUILD//8. This process cleans up the files and reduces their size, which helps minimize potential errors and shortens indexing time.
  - **Option 1: Full System Pack**
    - Selecting **Yes** will proceed with packing the entire system. This creates MUTEX files for each table, and the re-sync process can take considerable time.
  - **Option 2: Index Rebuild Only** (Default)
    - Selecting **No** will delete all the index files and rebuild the indexes, improving Collection-Master's performance without requiring SQL resyncing.
- However, with the introduction of SQL-Sync, running PROC REBUILD//8 and packing the database requires re-syncing the entire SQL database. Depending on the database's size, this re-sync process can take significant time.
- A new “ PACK ” methodology was introduced to address these challenges. This involves a program named PACK\_FILE/FIXPROG that optimizes one table at a time rather than the entire system. When this program is executed, it creates a file in the CLSINC\MUTEX folder, instructing SQL-Sync to use Bulk Insert for sessions instead of the default Incremental Sync until the table is fully rebuilt. Once the sync catches up, it returns to running Incremental Sync sessions as usual.
- Periodically packing the database can be valuable, as it will save space and time. However, you must balance the time this takes with the value.





# Using Time with SQL-SYNC

- `dbo_View_Paperless`: This view returns a well-formatted view of the Paperless file. It includes an example of converting time into a datetime format.
- `DATEADD(second, dbo.ACTIVE.TRANS_TIME / 100, dbo.ACTIVE.TRANS_DATE) AS DATE_TIME`
- `ufnsTime$`: This scalar function takes the time fields (expressed in 100ths of seconds since midnight).

- **Example**

```
SELECT TOP 1000 dbo.ufnsTime$(dbo.active.TRANS_TIME),  
DATEADD(second, dbo.ACTIVE.TRANS_TIME / 100, dbo.ACTIVE.TRANS_DATE) AS DATE_TIME  
FROM dbo.Active;
```



# Stp\_ReOrgIdx

- Reorganizes or rebuilds indexes in the database for optimized performance.
- Allow time for this script to complete. Some SQL Operations will be impacted.
- The script takes longer if it's been a while since you have maintained the database.
- Your DBA likely has maintenance scripts that perform similar tasks, but this is a good start.
- **Usage:**
  - EXEC [dbo].[Stp\_ReOrgIdx];

```
DECLARE @reorgLimit int = 5, @rebuildLimit int = 30
DECLARE @sql nvarchar(256), @numrows int

IF @reorgLimit > @rebuildLimit SET @reorgLimit = @rebuildLimit
DECLARE @scripts table (id int identity(1,1), runthis nvarchar(256), fragpct float)
insert into @scripts(runthis, fragpct)
SELECT 'ALTER INDEX [' + ind.name + '] ON ' + OBJECT_NAME(ind.OBJECT_ID) + CASE WHEN indexstats.avg_fragmentation_in_percent
BETWEEN @reorgLimit and @rebuildLimit THEN ' REORGANIZE;' WHEN indexstats.avg_fragmentation_in_percent > @rebuildLimit
THEN ' REBUILD;' END, indexstats.avg_fragmentation_in_percent
FROM sys.dm_db_index_physical_stats(DB_ID(), NULL, NULL, NULL, NULL) indexstats
INNER JOIN sys.indexes ind
ON ind.object_id = indexstats.object_id
AND ind.index_id = indexstats.index_id
WHERE indexstats.avg_fragmentation_in_percent > @reorgLimit
ORDER BY indexstats.avg_fragmentation_in_percent DESC
SET @numrows = @@ROWCOUNT

WHILE @numrows > 0
BEGIN
    SELECT @sql = runthis FROM @scripts WHERE id = @numrows
    EXECUTE sp_executesql @sql
    SET @numrows = @numrows - 1
END
END
```



# [Stp\_AccountAccountCard]

- Updates the AccountCard table with the latest data and recalculates relevant values.
- Notes:
  - This script requires that SETUP\_01 be populated (Make sure it's in the **dbo.[SyncSchedule]** table.
  - The AccountCard table is truncated before data is inserted to ensure the data is current.
- Example:
  - EXEC [dbo].[Stp\_Update\_AccountCard];

```
-- Set JMT Flags (what's included in JMT Amount), other flags
UPDATE #TEMP
SET Inc_Stat = CASE WHEN CHARINDEX('S',COLLORD)>0 THEN 0 ELSE 1 END
, Inc_CollCosts = CASE WHEN CHARINDEX('F',COLLORD)<=0 THEN 0 ELSE 1 END
, Inc_NRCosts = CASE WHEN CHARINDEX('N',COLLORD)>0 THEN 1 ELSE 0 END -- This flag is the opposite value of 1
, Jf_Suit = CASE WHEN UPPER(SUBSTRING(JMT_FLAGS,1,1))='N' THEN 1 ELSE 0 END
, Jf_Contract = CASE WHEN UPPER(SUBSTRING(JMT_FLAGS,2,1))='N' THEN 1 ELSE 0 END
, Jf_Stat = CASE WHEN UPPER(SUBSTRING(JMT_FLAGS,3,1))='N' THEN 1 ELSE 0 END
, Jf_Prejint = CASE WHEN UPPER(SUBSTRING(JMT_FLAGS,4,1))='N' THEN 1 ELSE 0 END
, Jf_Costs = CASE WHEN UPPER(SUBSTRING(JMT_FLAGS,5,1))='N' THEN 1 ELSE 0 END

UPDATE #TEMP
SET STAT_FEE = CASE WHEN STAT_FEE < 0 THEN 0 ELSE STAT_FEE END
, CONTRACT_FEE = CASE WHEN CONTRACT_FEE < 0 THEN 0 ELSE CONTRACT_FEE END

-- FnVCharges (calculate CALC_ORIG_CLAIM aka vCharges)
UPDATE #TEMP
SET VSTAT_DUE = convert(decimal(16,2),STAT_FEE - STAT_EARN - FEES_COLL)
, VCOSTS_DUE = convert(decimal(16,2),COST_EXP - COST_RECOVERED)
, VCONTRACT_DUE = convert(decimal(16,2),CONTRACT_FEE - CONTRACT_COLL)
, VCALC_PRIN_COLL = convert(decimal(16,2),PRIN_COLL - FEES_COLL - CONTRACT_COLL)
, VPRE_SUIT_PMTS = convert(decimal(16,2),MERCH_BEF + CASH_BEF + DP_PRE_SUIT_NF)
, VPOSTS_PMTS = convert(decimal(16,2),MERCH_POST + CASH_POST + DP_POST_SUIT_NF)
, VPOSTJ_PMTS = convert(decimal(16,2),MERCH_POST_JUDG + PAID_POST_JUDG + DP_POST_JUDG_NF)
```



# STP\_Create\_Date\_UPD\_Indexes

- Scans through SQL-SYNC tables that contain a Date\_UPD field that do not have an index.
- Creates an index to be leveraged by uncommitted changes.
- Example:
  - EXEC [dbo].[STP\_Create\_Date\_UPD\_Indexes];
- **Notes:**
  - This stored procedure will index many tables and may take a long time to execute.
  - Rerunning it will only create missing indexes.



# Stp\_Initialize\_Date\_EST\_Date\_UPD



- Using Changes and TRACKUSR, a Min\_Date and Max\_Date will be created for each claim.
- This #TEMP table will then update DATE\_EST & DATE\_UPD in MASTER, DEBTOR, EXTRA, DPHONE, DSERVE, LEGPROC, INFINITY, PROPERTY, INTERNAL.
- Note: Run this function to enable Uncommitted changes on old records with null values for DATE\_EST & DATE\_UPD.
- Example:
  - EXEC [dbo].[Stp\_Initialize\_Date\_EST\_Date\_UPD];
- Notes:
  - This stored procedure will update many records and may take a long time to execute.
  - Rerunning it will update records where DATE\_EST & DATE\_UPD are null.



To learn about upcoming or  
previous trainings:  
<https://vertican.com/mastermind/>



# MASTERMIND

SERIES





[www.VERTICAN.COM](http://www.VERTICAN.COM)