Migrating off the HP e3000: Been There, Done That

Steve LeBlanc, Ceridian, Steve.LeBlanc@ceridian.com
Victoria Shoemaker, Taurus Software, vas@taurus.com

Unless You Make It a Checkbook Experience, Migration Is Not for Beginners.

Overview

In June 1997, Ceridian began a journey to migrate our application from HP3000 to Unix. During this adventure, Ceridian learned many lessons, both the easy way and the hard way. This paper will share the journey and its lesson in hopes of making your pursuit easier.

Some background: Ceridian processes payroll taxes for both small and large organizations. This involves collecting information about hours worked, generating payroll checks, reporting to the various tax agencies (city, state, and federal), and ensuring compliance for the client to all taxing organizations. Failure to do the job right can result in severe penalties and interest costs for both Ceridian and the client.

In the beginning, we had two production HP3000 servers and two development servers. The two production servers were mirrors of each other. Netbase kept the two machines in sync. One was used for data entry and payroll process. The other was used for reporting and batch processing. Our business ran off these machines with a total of 500 business users accessing this application every day to meet their clients' needs.

First Date: June, 1997.
Blind Date, July 1998.
Drop-Dead Date: Y2K.
Actual Date, August, 1999.

In June 1997, Ceridian began planning for the year 2000 with the idea of completing the project within a year. After a flurry of meetings and conversations, it was decided that instead of just dealing with the year 2000 date issues, Ceridian would take the opportunity to solve other issues we faced. The issues that Ceridian hoped to address were:

- Get prepared for year 2000.
- Remove capacity limitations of the HP3000. Ceridian was sitting on the largest boxes available from HP and still needed more horsepower.
- Increase ability to handle ad-hoc reporting requests for data.
- Enable rapid change via open system technology.

- Move off the legacy application written in 1992 to meet Ceridian's business needs.

So in June of 1997, Ceridian set off to rewrite our application for open systems. The hardware that was chosen was HP/ux. The DBMS chosen was Oracle. The development language COBOL. And the job was broken into five major tasks:

- Converting the application code.
- Converting the JCL, QUIZ reports, and SUPRTOOL extracts.
- Creating of test environments.
- Data migration.
- Application testing and validation.

First, we had to decide who does what.

The conversion of the application code was contracted out to ICUBE. Their plan was to convert the COBOL code and leave the IMAGE and MPE calls in the code and handle them with "wrapper" code. That meant the code would still call DBGET, but a procedure would replace the functionality of that call with appropriate actions against the Oracle database. The task that faced ICUBE was not trivial. There were over 1000 programs to be migrated which amount to millions of lines of COBOL code. Although exact numbers are not known, it is thought that about a dozen programmers worked on this part of the project.

The conversion of the JCL, QUIZ reports and SUPRTOOL extracts fell to Ceridian. A team of four people was assigned to this task. A great portion of Ceridian's business was run in batch. Processing quarterly reports, producing W2, etc., was all done in batch. Getting this part of the job wrong could mean dire consequences for Ceridian.

Creation of the test environments and data migration was assigned to me, Steve. For my part, there were 210 datasets with over 400,000,000 rows of data spread over two machines that needed converting. The first step was to create test environments for the other developers and then the big conversion of all the production data. Most of my work was directed to these tasks, but as data was needed to test programs both for development and QA, I worked closely with all the groups and can speak to their issues from the 30,000-feet view.

The last bit, application testing and validation, was the largest part of the project and involved over 20 people working full-time. It was their responsibility to test each online module and ensure that the same results that occurred on the HP3000 application occurred on the new application.

Then we developed “The Plan.”

During the initial planning meetings, the following plan emerged for Ceridian's responsibilities:

- Choose a way to move the data.
- Map the data into Oracle tables.
- Determine which programs needed to be migrated.
- Create a test environment
    - Source test environment.
    - Target test environment.
- Design and develop the data movement process.
- Design and develop incremental data movement processes.
- Develop replacements for the QUIZ, JCL, and SUPRTOOL.
- Convert the COBOL code (ICUBE's responsibilities).
- Test, test, test…

The work was to begin in June, 1997, and finish July. 1998. (We actually completed in August of 1999.) Three teams were put together each with their own management.

How to move the data?

Ceridian's data structures were developed in 1974 along with the application. The plan for moving the data was to retain these structures with as few changes as possible. The only changes that would be considered were those necessitated by the new database's requirements. Extensive copylibs were used in the COBOL programs which overlayed structures on top of the IMAGE data structures. All dates were kept as 6 character data with a couple of magic dates values to represent various meanings to the application. Numeric data were kept as packed decimals which had to be converted to appropriate Oracle representations. Even though all the numeric data was stored as implied decimal data, the wrapper part of the application was to interpret this data and handle it.

The need for data was immediate. Therefore, our investigation into a method of moving the data started right away. We looked at the following methods:

- SUPRTOOL offered by Robelle (www.robelle.com).
- DBACCESS (out of business?).
- Writing programs in-house to move the data.
- BRIDGEWARE offered by Taurus Software (www.taurus.com).

SUPRTOOL

The data would be read from IMAGE using fast access methods by SUPRTOOL and outputted to a flat file. That flat file would FTPed to the new machine, and then the data would be loaded using SQLLoadSQLLoader (a utility provided by Oracle to do bulk loads of data). For our evaluation, we tried a couple of different scenarios in-house, as we already owned the tool.

If SQLLOADSQLLOADSQLLoad was used, one script for each different type of error would have to be created to load the data. Some of this would have to be written “on the fly”.

.

The advantages:

- Speed of downloading the data to a flat file and using SQLLoad to upload the information.
- No need to purchase any new tools. We already had a license for SUPRTOOL and SQLLoad came with the Oracle database.
- No need to learn a new tool.

The disadvantages:

- SUPRTOOL does not clean up the data, so data cleanup would have to be performed after each SQLLoad. In addition, placing the century in the middle of key fields was cumbersome.

We chose not use this product because of too much manual intervention. A person would have to monitor the progress of the SQLLoad and manually clean up the data, i.e.: fix dates, numbers, etc., and then rerun the process.

DBACCESS

The data would be read by DBACCESS and written directly to the Oracle tables. For our evaluation, we re-evaluated the tool in-house.

- The advantage: We already had the tool in house.

The disadvantages:

- Poor data mapping.

- No connectivity to the Oracle database.

We chose not use this product, because at the time, the product was not able to work between both platforms. Like SUPRTOOL, data had to be downloaded to a flat file with SQLLoad uploading the information to the Oracle Environment.

Writing Data Movement Programs In House

The data would be read by our in-house programs and written directly to the Oracle tables or flat files. These would then be FTP'd to the Unix box and uploaded using SQLLoad. For our evaluation, we wrote a sample program.

The advantages:

- We could programmatically cleanse the data using the program prior to writing the data to a flat file or directly to a Unix box.

- Data Mapping could also be performed in the code.

The disadvantages:

- Time limitations. This was one of the largest disadvantages. There were around 210 datasets that needed to be moved, this meant several very large programs

that were needed to be created. At the start of the project, it was determined that it would take one year to perform all the conversions and data movement. This was too long

- There was no easy way to develop the mirroring process that would be required.

We chose not use this method because a product was discovered that did everything we needed. What's more, creation of processes and testing of the processes was much simpler than creating and testing COBOL programs in the time frame we had. That product was BRIDGEWARE.

BRIDGEWARE

BRIDGEWARE read the source data using IMAGE intrinsics and wrote the data using intrinsic level access to Oracle. The product allowed bulk data movement between the two machines and provided the facility to capture changes to IMAGE environment as the data was change. We felt this would be helpful, as our initial load of the data was expected to run, 8 – 24 hour days split up on the weekends. This later turned out to be incorrect, but that is another story.

For our evaluation, we brought the product in-house and selected a couple of tables to take through the entire process. (See as an example the attachment, Table 1.)

The advantages:

- Ease of learning. The first day I received the DEMO version of this product, I wrote a process to move the data from the Image dataset to an Oracle table in less than 1 hour. Since we were already using Netbase for the mirroring of two HP3000's, BRIDGEWARE married directly to the process for the mirroring of the HP3000 to the HP9000. The tool easily connected to both machines and was able to cleanse the data before writing the information to the Oracle database.

- This product fully fitted into our requirement for moving the data from one environment to another. It wasn't until later that we decided to use the product to create the test environments during the process conversions.

The disadvantages:

- Need to learn a new tool. The learning curve, in fact, has proven to be very slight.

We chose BRIDGEWARE, because it fully fitted into our requirements for both data movement and mirroring. The second reason was the support. The support staff was

there and happy to help throughout the development process, regardless whether the question related to their product or data migration in general.

All of the evaluations were completed, and BRIDGEWARE was purchased by January 1998.

The saga begins…

So how do we start? We felt there would be many surprises in our data. There was no one person that understood all of the application. The data dated back to 1992. The application had undergone a number of major releases and changes to its original data structure. A new application to facilitate better reporting, performance or features might require that the data be changed.

We decided to start with a small piece of data and learn our lessons using the fewest number of tables as possible. This would enable us to get familiar with the new tool, the data, and the art of transforming. So what small set should we use? Well, the data could be grouped into six major groups:

- Data used by daily or weekly program (timesheet, weekly payroll processing, RJE)
- Data used for funding and deposit processing
- Data used for client inquiry
- Data used for creating quarterly reports and filings
- Data used for creating year-end reports and filings
- Data used for W2 processing

We elected the data used by the daily or weekly programs first. This was the smallest data group among the six. The daily conversion was also to be used as “proof of concept”.

For our first set of data, we began the process that we used for the remainder of the project:

- Create a test environment for testing.
- Map the data. (This becomes the specification for the routines to move the data.)
- Develop and test data movement procedures.

Creation of the test environment

A user/test committee chose the selected client id's to be used for the testing. A database with the subset of client-ids would be created in a static environment. All the other supporting files, other data files and programs, would be copied over so

that application access to the data would be available. Once the environment was created, a limited set of users would be granted access.

For our test environment, 171 client ids were chosen and only one quarter of the data base brought over. This resulted in a test environment of 32,000,000 sectors. We decided to refresh the environment periodically with current data environment, and this was done four times during the development period.

To create the test environment, we:

- Created a “sterile” database in the HP3000 where data can be pulled for before looks -- to several HP3000 and HP9000 databases and HP3000 image to HP3000 image -- as well as after looks -- HP3000 image to HP9000 Oracle.
- Programs; JCL; QUIZ were all placed on the HP3000 and HP9000 machines. Runs were performed on both, and scripts were run to verify that all tables and datasets were still in sync in the test environments.
- Each new test would pull from the sterile database and load data to the HP3000 and HP9000 test database prior to running the test in the processes.
- An “Image Library” was created which would trigger which warehouse load scripts to run to load the table data.
- If a process used only five datasets, those five datasets would be truncated and reloaded with the information prior to the test being run.

Mapping the data

The next step was to map the data. This process includes:

- Designing the target table.
- Mapping the source data to the target table.
- Describing any transformations that need to take place.
- How to handle errors.
- Any selection criteria to be used to subset the source data.

We were familiar with the data base and information (enough to be dangerous), and this made the creation of the scripts much easier.

We started with easiest datasets to be converted, and timed the data movement to get an idea of how long the process would take.

Some general decisions were arrived at quite quickly. We decided that we would only take data from 1/1/1993 forward. Data prior to that didn't have any of the supporting data (detail data) to support the aggregated data in the database.
The tables were to look pretty much as they did on the Image database system. All field names were the same with the exception of dashes being replaced as

underscores. Alpha numeric fields were replaced with varchar2; Numeric dates were replaced with dates; amounts replaced with numbers

Below is a table of these replacements;

Image Type	Oracle Type	Descriptions
All Alpha Numeric Fields	Varchar2	Exceptions with fields that contain special characters
All qtr/period fields; alpha numeric	Add 2 positions for Century;	Client_period 12345-009802 ; 12345-00199802
Alpha Numeric fields w/ special characters	Char	Datasets rdata, pr_hist_counter
Numeric date fields	Date	Oracle automatically ccyymmdd Magic date 12/31/99 will need to be set to 12/31/3000
Numeric (amount) fields	Number	All decimals are implied

Depending on the redefines used by the program, some tables required extra fields or separate tables to house that information (needed to verify that the redefines are still in use).

One main transformation was the addition of two characters to the qtr and/or period fields that were placed in the database. These fields were used to differentiate between year and quarters being processed for a given client id and/or tax id. The additional two characters held the century making the process Y2K compliant.

All programs and copylibs also needed to reflect this change. Below in the tables section is an example of the copylibs that were required for modifications.

```
*************************************************************
*                          WAGE-HEADER                      *
* 01-07-90 JRF ER382   Define second byte of WH-FLAGS field *
*                       for mag tape fileable state headers.*
*                                                           *
* 12/05/92 KT  ISR1-75 Add WH-WAGE-TRX-NUMX field           *
*                                                           *
* 01/08/92 MW  ISR1-63 Add WH-WAGE-TRX-NUMX field           *
*                                                           *
*************************************************************

 01 DB-WAGE-HEADER.
   05 WH-CLIENT-QTR-STATE.
```

```
      10 WH-CQS-CLIENT-ID           PIC X(08).
      10 WH-CQS-PERIOD              PIC X(6).
      10 WH-CQS-STATE               PIC X(02).
      10 WH-CQS-STATE-NUM REDEFINES WH-CQS-STATE   PIC 9(02).
   05 WH-STATE-QTR-EIN.
      10 WH-SQE-STATE               PIC X(02).
      10 WH-SQE-PERIOD              PIC X(6).
      10 WH-SQE-EIN                 PIC X(16).
   05 WH-WAGE-TRX-NUMX.
      10 WH-WAGE-TRX-NUM            PIC S9(9) COMP.
   05 WH-DATE-ADDCHG                PIC S9(8) COMP.
   05 WH-POST-DATE                  PIC S9(8) COMP.
*    05  WH-FLAGS                   PIC X(04).

   05 WH-FLAGS.
      10 WH-FLAGS-REPOST            PIC X(01).
      10 WH-FLAGS-MAG-FILEABLE                PIC X(01).
        88 WH-MAG-FILEABLE VALUE 'Y'.
      10 WH-FLAGS-3                 PIC X(01).
      10 WH-FLAGS-4                 PIC X(01).
```

One major issue that kept coming up was Invalid Numeric data in the field. This was due to invalid data being redefined from an alphanumeric field to a numeric field and them being placed into a comp-3 field in an image database. The handling was done with the use of a “TRY” function developed by Taurus. This function attempted to place numeric in a defined number oracle field. If the attempt failed the user had the choice of:

- Placing the error record in a file to be cleaned up at a later date.
- Setting the invalid number to 0 (zero) and writing the record.

I chose option two due to the date that the invalid numeric data was placed in. An assumption was made that a manual adjust has already been made to balance the information. Since the users were notified that this was how the information was going to be changed, any discrepancies would be adjusted for.

We also had three datasets mapped that contained different types of information depending on the Key being used. The key was 8 characters in length and the data area was 120 to 180 bytes in length, depending on the dataset. This needed to be converted into their own tables based on the key name. There were up to 26 different tables created from these 3 datasets.

Example:
Entity Key; data-160
PAYEE

This was converted to a table called payee-table with the data-160 byte field being separated into meaningful segments. This byte field contained, alpha numeric, numeric and comp3 defined information.

Cleansing of data:

- Date and numeric information was not very accurate; example – February does not have 30 days,
- In the old system, most changes to data was performed using an image db type tool, QTP COGNOS, most did not date check.
- Numeric amounts, redefines being performed in programs, alpha to numeric; ex 000-123456

Once these issues were identified, they were globally handled through the use of BRIDGEWARE user defined functions. (See Table 2 in attachments for an example.)

Mirroring Scripts

The BRIDGEWARE application has the ability to capture the changes as they are made to the IMAGE database or files. Those changes are then written to one or more files and then can be processed through a BRIDGEWARE script.(See Table 3 in attachments for an example.)

If these records are captured to a message file, you can continuously update the target environment throughout the day and keep the two environments "in sync". Another use is to turn the capture on and put the data into capture files and update once the initial load is complete. This method allows the users of application to continue using the HP3000 during the conversion. Once all the data has been converted, it can be "caught" up with data that was captured.

In our program, we grouped mirroring scripts in the same manner as the tables that were loaded, i.e., tax1, tax2, tax3, qtr1, etc.. As the transfer was completed, the mirroring scripts would get turned on for that group. This allowed for normal operation during the week. It also allowed us to migrate pieces of application and continue our on the new piece on the HPUX and still keep the pieces that had not been converted on the HP3000.

Conversion Facts:

- It took four weekends to complete the tasks of moving all the information from the Image database to Oracle, with the fourth weekend being the actual

changeover. Information was moved only on the weekend. Mirroring was used during the week to keep the databases in sync.

- Data was moved at about 1,000,000 rows/hr average. The actual speed varied from dataset to dataset depending on record length / clean up / number of columns of data.

- It took about 2 years to complete full conversion, including data migration and migration of the QUIZ, JCL, SUPRTOOL, and COBOL code.

- For our mirroring, we set up eight message files. A total of 900,000 records were captured per file per week. Operations were warned to call if over 400,000 records.

- Our original HP3000 hardware -- a 996 6-way primary 9X9 box (nova) shadow box -- used Netbase with EMC Symetric Disk Arrays.

- All data was spread out into 10 different databases, all these databases would be merged into one oracle data.

- System had ½ gig of memory.

- Applications are single threaded.

- 1st step was locating all date, numeric, and quarter/period fields.

- The qtr/period fields were usually truncated with client_id, tax_code, tax_id and/or flags throughout the system.

- Most were placed in copylibs, some were hard coded in programs

What did we do right?

- Management was behind us all the way! We had management's support for the very beginning all the way through the end of the project. This lead to resources being available and road blocks being removed.

- The team was committed 100%! If we were to chose the single factor that played most to the success of the project it would have to be **unwavering team commitment**. In migrations, there are going to be problems. In fact, there are going to be problems that you can't even imagine at the outset of the project. For us, perseverance paid off. Despite what were at times, overwhelming obstacles

(like our tape drive issues), the dedication, tenacity, and even stubbornness of team members led to our success.

- Testing, testing, and re-testing! We knew from the beginning that testing was going to be the key to a successful implementation. Consequently, we developed an exhaustive test plan. Each major deliverable had a corresponding set of acceptance criteria and then we test all those items and more.

- We didn't panic. There were a number of team members on the team that had been through migrations before. An implementation will only fail if you let it.

What would we do differently?

- Our team structure was not set up correctly. We were using a matrix organization that doesn't work for these large projects. What is needed is a dictator. Okay, maybe not that bad, but definitely clear roles and responsibilities, clear authority, a communications plan which must be developed, reworked and enforced.

- We didn't have project management personnel who had experience with projects of this size. My guess is most HP3000 organizations don't. This resulted in a number of bad things (no qualified dictator, too much reliance on the code translator and unrealistic schedule). You may want to consider hiring a contractor with this type of experience to balance your lack of skills. You should have your internal dictator shadow this person and integrate the process into your internal process throughout your organization.

- As all the code was being migrated by a 3rd party vendor, we relied too heavily on their testing. This caused rework, delays, and sometimes an uneven distribution of the workload. Often, the work would stop until there was a fix in place.

- The schedule was not realistic and the scope was too big. There were signs early on that there were problems meeting deadlines. Trust me, you **can't** make up time. If you are off schedule at the beginning of the schedule, it will only get exponentially worse towards the end of the project. Confront these problems quickly, do not deny their impact.

Tips and hints for you.

- It is all in the planning. Design and forethought will enable you to plan your resources correctly. Start at the top. Make decisions that you can live with:

- What you are going to move.
- What kinds of data structures you are going to have in place.
- General guidelines about data (i.e.: is it more important to have dirty data and the data in the new database or is it more important to have clean data and those things which are not clean will not be there).
- A plan for how you are going to make sure that everything is working.

Once you have this, develop a high-level plan and double the times. Remember you can only count on 30-36 hours a week of productive work from any given individual.

- Once you have a high-level plan, plan the data move. The mapping document will become your specification for the move. This mapping document will detail out the movement, the kinds of errors and how they are handled. This will help you develop an appropriate test environment.

- A perfect test environment is just enough data to test everything. The environment should be static, so if you run the test 40 times you should get the same results 40 times. Creating the test environment may be hard, but well worth the effort.

- Divide up your team. Each team leader should be meeting with their team each day and dealing with problem. You should be meeting with team leads each day to ensure that everything stays on schedule. Your job at this point to get answers quickly. Make sure that you are always working on the most important task and that you don't get bogged down. It is also essential to make a good decision and not to spend time revisiting issues that have been thoroughly considered.

- Devise a communication plan. There is nothing more annoying that to have your whole development team ready to run the "big test" this weekend only to find out the Oracle DBA team is going to use this weekend for another "big test" and the environment is not going to be available. If teams have enough notice, they can plan accordingly. If they don't know, this will lead to huge morale issues. Remember happy programmers (or fill in the blank) are productive programmers.

- Version control is essential. Running the "big test" with the wrong programs causes rework. Rework is death to the schedule. For every coding mistake: you have programming time, programming testing time, and QA test time. Everything has to be revalidated.

- Make sure to put time in for running your tests against the "real" database. You might find that things will take longer or shorter or have issues you haven't

discovered yet. In our case, the loads took way shorter than expected, and we were able to move the schedule around significantly. Without this knowledge, we would have had whole teams doing **nothing** except for waiting for time to pass!

- Don't forget to put time in for the last 80% of the work. Getting things into production once the process is finished is important. Remember you aren't done until you are done and off the old system.

This is not rocket science.
Just close!

Despite lack of experience with migrations and of information from others who had gone through the same journey, Ceridian prevailed. In the course of the project, we found the right tools, the right people, and the right methodologies to get the job done. No, we did not finish on the original completion date, this time. But next time (and for Pete's sake, I hope that isn't too soon), we can.

Table 1:

LOCATED DATE FIELDS AND OTHER ISSUES IN THE DATABASES

⇒ STSTAX:

SET NAME	VARIABLE	TYPE	FORMAT	COPYLIB / VARIABLE	COMMENT
BATCH-MASTER				**TFBCHFIL.COPYLIB**	
	DATE-ADDCHG	S9(6) COMP	YYMMDD	BCH-DATE-ADDED	
	ADJUST-FLAGS • QTR-PERIOD	X(8) • X(3) pos 6-8	YYQ	BCH-ADJUST-FLAGS BCH-QTR-PERIOD	
	LIABILITY-DATE	S9(6) COMP	YYMMDD	BCH-LIABILITY-DATE	
	POST-DATE	S9(6) COMP	YYMMDD	BCH-POST-DATE	redefine of liability-date
CHECK-TRANS				**TFCKTFIL.COPYLIB**	
	DUE-DATE	S9(6) COMP	YYMMDD	CHKTRANS-DUE-DATE	
	LIABILITY-DATE	S9(6) COMP	YYMMDD	CHKTRANS-LIABILITY-DATE	
	CHECK-DATE	S9(6) COMP	YYMMDD	CHKTRANS-CHECK-DATE	
	POST-DATE	S9(6) COMP	YYMMDD	CHKTRANS-POST-DATE	
	DEPOSIT-DATE	S9(6) COMP	YYMMDD	CHKTRANS-DEPOSIT-DATE	
ENTITY	(key)				**EACH KEY TYPE IS LOCATED IN A SEPARATE COPYLIB** ENTITY-DATA x(140) actually holds binary, dates and characters
	AOCTBL	9(6) pos 1-6	YYMMDD	**QSAOCFIL.COPYLIB /** AOCTBL-DEP-DATE	
		9(6) pos 11-16	YYMMDD	LAST-UPD-DATE	

RECEIVED	ASSOC	N/A		**TFASSOC.COPYLIB**	
RECEIVED	BANKID	N/A			no copylib found for this key, data displays no dates
RECEIVED	BKXF	X(2) pos 1-80		**TFBKXF.COPYLIB**	occurs 40 times array
RECEIVED	BYGCTL	N/A		**TFBYGCTL.COPYLIB**	
	CNVBID	N/A		**TFCNVBID.COPYLIB**	
	COMFLT	X(4) pos 7-12	YYMMbb	**TFCOMFLT.COPYLIB /** COMFLT-COMM-PERIOD	redefine of 2nd half of key
		9(6) COMP pos 114-117	YYMMDD	COMFLT-STATUS-DATE	
		9(6) COMPpos 118-121	YYMMDD	COMFLT-LAST-UPD-DATE	
	COMPN2	S9(6) COMP pos 40-45	YYMMDD	**TFCOMPNY.COPYLIB /** COMPN2-STATUS-DATE	
		S9(6) COMP pos 46-51	YYMMDD	COMPN2-LAST-UPD-DATE	
	COMPNY	N/A		**TFCOMPNY.COPYLIB**	
	CSVREP	N/A		**TFCSVREP.COPYLIB**	
	DECTRL	N/A		**TFDECTRL.COPYLIB**	
	EDTMSG	9(6) pos 32-37	YYMMDD	**QSENTFIL.COPYLIB** / EDTMSG-STATUS-DT	
		9(6) pos 38-43	YYMMDD	LAST-UPD-DATE	
	FNDCTL	N/A		**TFFNDCTL.COPYLIB**	
	FORMSG	N/A		**TFFORMSG.COPYLIB**	
	HOLDAY	9(2) pos 7-12	YYbbbb	**TFHOLDAY.COPYLIB /** HOLIDAY-TABLE-YEAR	
		S9(6) pos 1-96	YYMMDD	HOLIDAY-DATE	array 16 x 6
	INALED	9(6) pos 3-8	YYMMDD	**TFINALED.COPYLIB /** INALED-CLT-LED-EXT-DATE	
		9(12) pos 9-108	YYMMDD	INALED-SET-BEG-DATE, INALED-SET-END-DATE	occurs 9 times
		9(6) pos112-118	YYMMDD	INALED-LAST-UPD-DATE	
	NULREC				unknown
	PAYEE	9(6) pos 194-199	YYMMDD	**TFPAYEE.COPYLIB /**	

				PAYEE-UPD-DATE	
	PHSSTA	9(6) pos 42-47	YYMMDD	**TFPHSSTA.COPYLIB /** PHSSTA-STATUS-DATE	
		9(6) pos 120-125	YYMMDD	PHSSTA-LAST-UPD-DATE	
	PRHSTA	9(6) pos 42-47	YYMMDD	**TFPRHSTA.COPYLIB /** PRHSTA-STATUS-DATE	
		9(6) pos120-125	YYMMDD	PRHSTA-LAST-UPD-DATE	
	PRSOFT			**TFPRSOFT.COPYLIB**	N/A
	PRTCOM	9(6) pos 42-47	YYMMDD	**TFPRTCOM.COPYLIB** / PRTCOM-STATUS-DATE	
		9(6) pos 120-125	YYMMDD	PRTCOM-LAST-UPD-DATE	
	PRTMSG			**TFPRTMSG.COPYLIB**	N/A
	RCVPRM			**TFRCVPRM.COPYLIB**	N/A
	RETFRE				no copylib found
	SALESM				
	STATEA	9(6) pos 32-37	YYMMDD	**TFSTATEA.COPYLIB /** STATEA-STATUS-DATE	
		9(6) pos117-122	YYMMDD	STATEA-LAST-UPD-DATE	
	STATEN	9(6) pos 32-37	YYMMDD	**QSENTFIL.COPYLIB /** STATEN-STATUS-DT	
		9(6) pos 127-132	YYMMDD	LAST-UPD-DATE	
	STSREP			**QSENTFIL.COPYLIB**	N/A
	TAPIO1	S9(6) pos 63-68	YYMMDD	**TFTAPIO1.COPYLIB /** TAPIO1-TAPE-APPROVAL-DATE	
	TAPIO2			**QSTAPIO2.COPYLIB**	N/A
	USERNO	9(6) pos 42-47	YYMMDD	**TFUSERNO.COPYLIB /** USERNO-STATUS-DATE	
		9(6) pos 119-124	YYMMDD	USERNO-LAST-UPD-DATE	
	W2CODE	S9(6)COMP pos 1-4	YYMMDD	**QSENTFIL.COPYLIB** / LAST-UPD-DATE	
		S9(6) COMP pos 36-39	YYMMDD	W2-DUE-DATE	
	WAGE			**TFXFCDTP.COPYLIB**	N/A
	XFERCD			**TFXFRPRM.COPYLIB**	N/A
	XFRPR2				

	XFRPRM	9(6) pos 122-127	YYMMDD	**TFXFRPRM.COPYLIB** / XFER-LAST-UPD-DATE	
TAX-CODE				**TFTAXFIL.COPYLIB**	
	TAX-CODE-DESC • FORMAT-TYPE	X(1) POS 10		TAX-DATE-FORMAT-TYPE	Contains format date type flag, 'G' = MMDDYY; 'Q' = nQYY
	TAX-LST-UPD-DATE	S9(06) COMP	YYMMDD	TAX-LST-UPD-DATE	
	ACTIVE-FROM-DATE	S9(06) COMP	YYMMDD	TAX-ACTIVE-FROM-DATE	
	ACTIVE-TO-DATE	S9(06) COMP	YYMMDD	TAX-ACTIVE-FROM-DATE	
HIST-BATCH-MSTR				**TFHBMFIL.COPYLIB**	
	CLT-QTR-FMT-TYPE CLIENT QUARTER FORMAT TYPE	X(14) X(8) pos 1-8 9(4) pos 9-12 X(1) pos 13 X(1) pos 14	YYQQ	HIST-BAT-MSTR-KEY	
FLOAT-FACTORS				**TFFLTFIL.COPYLIB**	
	DAILY-FACTOR	V9(06) COMP		DAILY-FACTOR	array 366 times
CHECK-TRANS-DTL				**TFCTDFIL.COPYLIB**	
	LIABILITY-DATE	S9(6) COMP	YYMMDD	CHKTRN-DTL-LIABILITY-DATE	
	POST-DATE	S9(6) COMP	YYMMDD	CHKTRN-DTL-POST-DATE	
CLIENT				**TFCLTFIL.COPYLIB**	
	DATE-ADDCHG	S9(6) COMP	YYMMDD	CLT-DATE-ADDCHG	
	TERM-DATE	S9(6) COMP	YYMMDD	CLT-TERM-DATE	
	SCHED-START	S9(6) COMP	YYMMDD	CLT-SCHED-START	
	ACTUAL-START	S9(6) COMP	YYMMDD	CLT-ACTUAL-START	

	DATE-FINAL-WAGES	S9(6) COMP	YYMMDD	CLT-DATE-FINAL-WAGES	
	HIST-FILES	X(16)		CLT-HIST-FILES	array 7 times
	(1)				
	LAST-UPD-DATE	S9(6) COMP pos 1-4	YYMMDD	CLT-LAST-UPDATE-DATE	
	LAST-UPD-TIME	S9(6) COMP pos 5-8	HHMMSS	CLT-LAST-UPDATE-TIME	
	LAST-UPD-USER	X(8) pos 9-16		CLT-LAST-UPDATE-USER	
	(2)				
	CUST-SERV-REP	X(3) pos 1-3		CLT-CUST-SERV-REP	
	EST-PAYROLL-OPT	X(1) pos 4		CLT-EST-PAYROLL-OPTION	
				CLT-EST-PAYROLL-AMOUNT	
	EST-PAYROLL-AMT	S9(9)V99 COMP-3 pos 5-10		CLT-EST-PAYROLL-EFF-DATE	
				CLT-ACH-CODE	
	EST-PAYROLL-EFF-DATE	S9(6) COMP pos 11-14	YYMMDD	CLT-FAX-PHONE	
	CLT-ACH-CODE			CLT-FAX-ALLOW-FLAG	
	(3)	X(2) pos 15-16		CLT-MAJ-ACCT-FLAG	
	FAX-PHONE				
	FAX-ALLOW-FLAG	X(14) pos 1-14		CLT-PR-VAR-PCT-ALLOWED	
	MAJ-ACCT-FLAG	X(1) pos 15		CLT-AVG-PAYROLL-AMT	
	(4)	X(1) pos 16		CLT-DATA-COLL-REP	
	PAYROLL-VAR-PCT-ALLOWED			CLT-CREATE-PAYROLL-FLAG	
				CLT-MID-Q-Y-STATUS-CODE	
	AVE-PAYROLL-AMT	S9(4) COMP pos 1-2		CLT-W2-FEE-STATUS	
	DATA-COLL-REP	S9(13)V99 COMP-3, pos 3-10		CLT-W2-PROCESSED	
	CREATE-PAYROLL-FLAG	X(2) pos 11-12		CLT-W2-FILE-YR	
	MID-Q-Y-STATUS-CODE	X(1) pos 13		CLT-W2-FILE-FED	
	W2-FEE-STATUS			CLT-W2-FILE-STATE	
	W2-PROCESSED	X(1) pos 14		CLT-INSTALL-USER-ID	array 4 times
	(5)	9(1) pos 15		CLT-INSTALL-DATE	array 4 times
	W2-CLIENT-YEAR	X(1) pos 16		CLT-STATUS-DATE	array 4 times
	W2-FILE-FED				
	W2-FILE-STATE	9(02) pos 1-8		CLT-W2-FILE-LOCAL	

	(6) INSTALL-USER-ID INSTALL-DATE STATUS-DATE (7) W2-FILE-LOCAL STATUS-FED STATUS-STATE POA- GRANDFATHER PAY-TRACKING- FLAG FILLER	X(1) pos 9-12 X(1) pos 13-16 X(8) pos 1-8 S9(6) COMP pos 9-12 S9(6) COMP pos 13-16 X(1) pos 1 - 4 S9(6) COMP pos 5-8 S9(6) COMP pos 9-12 X(1) pos 13 X(1) pos 14 X(2) pos 15-16	 YYMMDD YYMMDD	W2-STATUS-FED W2-STATUS-STATE CLT-POA-GRANDFATHER CLT-PAY-TRACKING-FLAG FILLER	 array 4 times
	LAST-BILL-DATE	S9(6) COMP	YYMMDD	CLT-LAST-BILL-DATE	
	BILL-LAST-PERIOD-START	S9(6) COMP	YYMMDD	CLT-BILL-LAST-PERIOD-START	
	BILL-LAST-PERIOD-END	S9(6) COMP	YYMMDD	CLT-BILL-LAST-PERIOD-END	
	KIT-RECVD-DATE	S9(6) COMP	YYMMDD	CLT-KIT-RECVD-DATE	
	WORKSITE-BEGIN	S9(6) COMP	YYMMDD	CLT-WORKSITE-BEGIN	
	WORKSITE-END	S9(6) COMP	YYMMDD	CLT-WORKSITE-END	
CLIENT-TAX-CODE				**TFCTXFIL.COPYLIB**	
	TAX-EST-CHG-DATE	S9(6) COMP	YYMMDD	CLT-TAX-EST-CHG-DATE	
	LAST-UPD-DATE	S9(6) COMP	YYMMDD	CLT-TAX-LAST-UPDATE-DATE	
	STATUS-DATE	S9(6) COMP	YYMMDD	CLT-TAX-STATUS-DATE	
	TAX-AGENCY-SPECIFIC CTAS-DATE-1 CTAS-DATE-2	X(20) 9(6) pos 1-6 9(6) pos 7-12	 YYMMDD YYMMDD	CLT-TAX-AGENCY-SPEC	
MAG-TAPE-FILE				**TFMAGFIL.COPYLIB**	

	CPN-APPLY-DATE	S9(6) COMP	YYMMDD	MAG-CPN-APPLY-DATE	
	CPN-APPROVE-DATE	S9(6) COMP	YYMMDD	MAG-CPN-APPROVE-DATE	
	CPN-DISAPPR-DATE	S9(6) COMP	YYMMDD	MAG-CPN-DISAPPR-DATE	
	CPN-TERM-DATE	S9(6) COMP	YYMMDD	MAG-CPN-TERM-DATE	
	RET-APPLY-DATE	S9(6) COMP	YYMMDD	MAG-RET-APPLY-DATE	
	RET-APPROVE-DATE	S9(6) COMP	YYMMDD	MAG-RET-APPROVE-DATE	
	RET-DISAPPR-DATE	S9(6) COMP	YYMMDD	MAG-RET-DISAPPR-DATE	
	RET-TERM-DATE	S9(6) COMP	YYMMDD	MAG-RET-TERM-DATE	
	WGE-APPLY-DATE	S9(6) COMP	YYMMDD	MAG-WGE-APPLY-DATE	
	WGE-APPROVE-DATE	S9(6) COMP	YYMMDD	MAG-WGE-APPROVE-DATE	
	WGE-DISAPPR-DATE	S9(6) COMP	YYMMDD	MAG-WGE-DISAPPR-DATE	
	WGE-TERM-DATE	S9(6) COMP	YYMMDD	MAG-WGE-TERM-DATE	
COMMENTS				**TFCMTFIL.COPYLIB**	
	CMT-KEY QTR-CMT-LITERAL QTR-CMT-PER-YY QTR-CMT-PER-Q QTR-CMT-NOTE-CODE	X(8) X(1) pos 1 X(2) pos 2-3 X(1) pos 4 X(4) pos 5-8	 YY 	CMT-KEY QTR-CMT-LITERAL QTR-CMT-PERIOD-YY QTR-CMT-PERIOD-Q QTR-CMT-NOTE-CODE	
LEDGER-TRX				**TFLTXFIL.COPYLIB**	
	DATE-ADDED	S9(6) COMP	YYMMDD	LED-TRX-DATE-ADDED	
	LIABILITY-DATE	S9(6) COMP	YYMMDD	LED-TRX-LIABILITY-DATE	
	POST-DATE	S9(6) COMP	YYMMDD	LED-TRX-POST-DATE	
	CHECK-DATE	S9(6) COMP	YYMMDD	LED-TRX-CHECK-DATE	
	HIST-POST-DATE	S9(6) COMP	YYMMDD	LED-TRX-HIST-POST-DATE	
	CURR-PERIOD	X(4)	YYQQ	LED-TRX-CURR-PERIOD	assumed, No data

					in this field
LIABILITY-HDR				**TFLBHFIL.COPYLIB**	
	DATE-ADDCHG	S9(6) COMP	YYMMDD	LBH-DATE-ADDCHG	
	LIABILITY-DATE	S9(6) COMP	YYMMDD	LBH-LIABILITY-DATE	
	DUE-DATE	S9(6) COMP	YYMMDD	LBH-DUE-DATE	
	PERIOD-END	S9(5) COMP	YYQNN	LBH-PERIOD-END	
	CHECK-DATE	S9(6) COMP	YYMMDD	LBH-CHECK-DATE	
	LAST-UPDATE-DATE	S9(6) COMP	YYMMDD	LBH-LAST-UPDATE-DATE	
LIABILITY-DET				**TFLBDFIL.COPYLIB**	
	LIABILITY-DATE	S9(6) COMP	YYMMDD	LBD-LIABILITY-DATE	
PAYROLL-HEADER				**TFPRHFIL.COPYLIB**	
	ENTRY-NO ENTRY-NO-REAL ENTRY-NO-COMP	X(2) 9(2) S9(4) COMP	XX 99 -9999	PRH-ENTRY-NO PRH-ENTRY-NO-REAL PRH-ENTRY-NO-COMP	 redefine to numeric redefine to binary
	DATE-ADDCHG	S9(6) COMP	YYMMDD	PRH-DATE-ADDCHG	
	POST-DATE	S9(6) COMP	YYMMDD	PRH-POST-DATE	
	LIABILITY-DATE FUNDS-RCVD-DATE	S9(6) COMP S9(6) COMP	YYMMDD YYMMDD	PRH-LIABILITY-DATE PRH-FUNDS-RCVD-DATE	 redefine
	FUNDS-XFER-POST-DATE	S9(6) COMP	YYMMDD	PRH-FUNDS-XFER-POST-DATE	
	EXP-DUE-DATE EXP-DUE-DATE-NUM	X(6) 9(6)	YYMMDD	PRH-EXP-DUE-DATE PRH-EXP-DUE-DATE-NUM	 redefine
	STATUS-DATE	S9(6) COMP	YYMMDD	PRH-STATUS-DATE	
	BATCH-KEY BATCH-DATE BATCH-SEQ-NO RJE-BATCH-YEAR RJE-BATCH-FILE	X(10). 9(06) pos 1-6 9(04) pos 7-10 9(02) pos 1-2 X(08) pos 3-10	 YYMMDD YY	PRH-BATCH-KEY PRH-BATCH-DATE PRH-BATCH-SEQ-NO PRH-RJE-BATCH-YEAR PRH-RJE-BATCH-FILE	 redefine

	LAST-UPD-DATE	S9(6) COMP	YYMMDD	PRH-LAST-UPD-DATE	
PAYROLL-DETAIL				**TFPRDFIL.COPYLIB**	
	CLIENT-ENTRY-NO CLIENT-ID ENTRY-NO ENTRY-NO-REAL ENTRY-NO-COMP	X(10) X(8) pos 1-8 X(2) pos 9-10 9(2) S9(4) COMP		PRD-CLIENT-ENTRY-NO PRD-CLIENT-ID PRD-ENTRY-NO PRD-ENTRY-NO-REAL PRD-ENTRY-NO-COMP	 redefine to numeric redefine to binary
	DUE-DATE LIABILITY-PERIOD	S9(6) COMP S9(6) COMP	YYMMDD	PRD-DUE-DATE PRD-LIABILITY-PERIOD	 redefine
PAYROLL-HIST				**TFPHSFIL.COPYLIB**	
	DATE-ADDED	S9(6) COMP	YYMMDD	PAY-HIST-DATE-ADDED	
	STATUS-DATE	S9(6) COMP	YYMMDD	PAY-HIST-STATUS-DATE	
	DATE-RECEIVED	S9(6) COMP	YYMMDD	PAY-HIST-DATE-RECEIVED	
	EXP-DUE-DATE	X(6)	YYMMDD	PAY-HIST-EXP-DUE-DATE	
	LAST-UPD-DATE	S9(6) COMP	YYMMDD	PAY-HIST-LAST-UPD-DATE	
	PAYROLL-DATE LIABILITY-DATE	X(6) 9(6)	YYMMDD	PAY-HIST-PAYROLL-DATE PAY-HIST-LIABILITY-DATE	 redefine
	LAST-CHECK-DATE	S9(6) COMP	YYMMDD	PAY-HIST-LAST-CHECK-DATE	
	TSR-ENTRY-NBR	X(2)		PAY-HIST-TSR-ENTRY-NBR	assume binaries held
QTR-RECON				**TFQTRFIL.COPYLIB**	
	CLIENT-PER ID CLT-PER	X(12) X(8) pos 1-8 X(4) pos 9-12	 YYQQ	QTR-CLIENT-PER QTR-ID QTR-CLT-PER	
	PERIOD	X(4)	YYQQ	QTR-PERIOD	
	DATE	S9(6) COMP	YYMMDD	QTR-DATE	
	TAX-CODE	X(4)		QTR-TAX-CODE	array 12 times
	TAX-WITHHELD	S9(9)V99 COMP-3		QTR-TAX-WITHHELD	array 12 times
	TAX-DUE	S9(9)V99 COMP-3		QTR-TAX-DUE	array 12 times

	NBR-QTRS	S9(2) COMP		QTR-NBR-QTRS	array 3 times
	NBR-TYPES	S9(3) COMP		QTR-NBR-TYPES	array 3 times
	SETUP-PRICE	S9(2)V99 COMP		QTR-SETUP-PRICE	array 3 times
	NBR-POSTINGS	S9(3) COMP		QTR-NBR-POSTINGS	array 3 times
	POSTING-PRICE	S9(2)V99		QTR-POSTING-PRICE	array 3 times
	NBR-RETURNS	S9(3) COMP		QTR-NBR-RETURNS	array 3 times
	RETURN-PRICE	S9(2)V99 COMP		QTR-RETURN-PRICE	array 3 times
	CLT-ID-PROC	PIC X(8)		QTR-CLT-ID-PROC	array 3 times
	CMNT-ID	PIC X(2)		QTR-CMNT-ID	array 4 times
REPOST-HIST-HDR				**TFRPHFIL.COPYLIB**	
	HDR-QUARTER	X(4)	YYQQ	REP-HDR-QUARTER	
	HDR-QTR	X(4)	YYQQ	REP-HDR-QTR	
	DATE-ADDCHG	S9(6) COMP	YYMMDD	REPOST-DATE-ADDCHG	
	DATE-POSTED	S9(6) COMP	YYMMDD	REPOST-DATE-POSTED	
REPOST-HIST-DTL				**TFRPDFIL.COPYLIB**	
	DTL-QUARTER	X(4)	YYQQ	REP-DTL-QUARTER	
	DTL-QTR	X(4)	YYQQ	REP-DTL-QTR	
TAX-ID-CHG-TRX				**TFCTTFIL.COPYLIB**	
	DATE-ADDCHG	S9(6) COMP	YYMMDD	CTT-DATE-ADDCHG	
	POST-DATE	S9(6) COMP	YYMMDD	CTT-POST-DATE	
	QTR-PERIOD	X(4)	YYQQ	CTT-QTR-PERIOD	
CHECK-MAINT-TRX				**TFCMHFIL.COPYLIB**	
	DATE-ADDED	S9(6) COMP	YYMMDD	CMH-DATE-ADDED	
	CHECK-DATE	S9(6) COMP	YYMMDD	CMH-CHECK-DATE	
	LIABILITY-DATE	S9(6) COMP	YYMMDD	CMH-LIABILITY-DATE	
	DUE-DATE CHECK-NO-2	S9(6) COMP S9(6) COMP	YYMMDD YYMMDD	CMH-DUE-DATE CMH-CHECK-NO-2	 redefine

	POST-DATE	S9(6) COMP	YYMMDD	CMH-POST-DATE	
	APPROVAL-DATE	S9(6) COMP	YYMMDD	CMH-APPROVAL-DATE	
	DEPOSIT-DATE	S9(6) COMP	YYMMDD	CMH-DEPOSIT-DATE	
CHECK-MAINT-DTL				**TFCMDFIL.COPYLIB**	
	LIABILITY-DATE	S9(6) COMP	YYMMDD	CMD-LIABILITY-DATE	
TAX-PRINT-XREF				**TFTCPFIL.COPYLIB**	
	QTR-PERIOD	9(4)	YYQQ	TCP-QTR-PERIOD	
BANK-TRANSFERS				**TFBFXFIL.COPYLIB**	
	DATE-ADDCHG	S9(6) COMP	YYMMDD	BANK-XFER-DATE-ADDCHG	
	POST-DATE	S9(6) COMP	YYMMDD	BANK-XFER-POST-DATE	
	LIABILITY-DATE	S9(6) COMP	YYMMDD	BANK-XFER-LIABILITY-DATE	
	PRH-ENTRY-NO	9(2)		BANK-XFER-PRH-ENTRY-NO	also holds binary num
	FUNDS-XFER-DATE	S9(6) COMP	YYMMDD	BANK-XFER-FUNDS-XFER-DATE	
	PROCESS-DATE	S9(6) COMP	YYMMDD	BANK-XFER-PROCESS-DATE	
MAG-CPN-EX				**TFMCEFIL.COPYLIB**	
	DEP-DATE	9(6)	YYMMDD	MTCIDDT-DEP-DATE	
	LIABILITY-DATE	S9(6) COMP	YYMMDD	MCE-LIABILITY-DATE	
	LIAB-PERIOD-MMYY	X4	MMYY	MCE-LIABILITY-PERIOD	
	LAST-UPD-DATE	S9(6) COMP	YYMMDD	MCE-LAST-UPD-DATE	
BANK-HISTORY				**TFBHSFIL.COPYLIB**	
	DATE-ADDED	S9(6) COMP	YYMMDD	BHS-DATE-ADDED	
	XPECT-XFER-DATE	S9(6) COMP	YYMMDD	BHS-XPECT-XFER-DATE	
	QTR-PERIOD	9(4)	YYQQ	BHS-QTR-PERIOD	
	BANK-DATE	S9(6) COMP	YYMMDD	BHS-BANK-DATE	

PAYROLL-DATES					
	PAYROLL-DATE	S9(6) COMP	YYMMDD		
	CALC-DATE	9(6)	YYMMDD		
PRINT-QUEUE					
	QTR-PERIOD	X(4)	YYQQ		
	DATE-ADDCHG	S9(6) COMP	YYMMDD		
ENTITYX					
	ENTITYX-DATA	X()			
	PAYEE (key)	pos 197-202	YYMMDD		
PRPROC-CODE-XREF					
	LAST-UPD-DATE	S9(6) COMP	YYMMDD		
	STATUS-DATE	S9(6) COMP	YYMMDD		
CHECK-REG-TRX					
	DATE-ADDED	S9(6) COMP	YYMMDD		
	POST-DATE	S9(6) COMP	YYMMDD		
	DEPOSIT-DATE	S9(6) COMP	YYMMDD		
MAG-TAPE-HIST					
	DEPOSIT-DATE	9(6)	YYMMDD		
	TAPE-RETURN-DATE	S9(06)COMP	YYMMDD		
	DATE-ORIG-CREATED	S9(06) COMP	YYMMDD		
	DATE-LAST-CREATED	S9(06) COMP	YYMMDD		
	DATE-EXTRACT-CREATED	S9(06) COMP	YYMMDD		
	LAST-UPD-DATE	S9(06) COMP	YYMMDD		

OPEN-ITEM-HEADER					
	POST-DATE	S9(6) COMP	YYMMDD		
	PERIOD-START-DATE	S9(6) COMP	YYMMDD		
	PERIOD-END-DATE	S9(6) COMP	YYMMDD		
	DUE-DATE	S9(6) COMP	YYMMDD		
	COLLECTION-DATA CREATING-PGM COLLECTION-DATE COLLECTION-TYPE	X(8) X(2) pos 1-2 S9(6) COMP pos 3-6 X(2) pos 7-8	YYMMDD		
	ORIG-PRINT-DATE	S9(6) COMP	YYMMDD		
OPEN-ITEM-DETAIL					
	LIABILITY-DATE	S9(6) COMP	YYMMDD		
FEE-HIST-DTL					
	BILLING-PERIOD	9(4)	YYMM		
	DATE-ADDCHG	S9(6) COMP	YYMMDD		
	LIABILITY-DATE	S9(6) COMP	YYMMDD		
FEE-HIST-SUM					
	BILLING-PERIOD	9(4)	YYMM		
	LAST-UPDATE-DATE	S9(6) COMP	YYMMDD		
	TERM-DATE	S9(6) COMP	YYMMDD		
BALANCE-TOTALS					
	CLIENT-TAX-ACCT-YEAR	X(16)			

	CLIENT-ID TAX-CODE ACCOUNT-TYPE YEAR	X(08) pos 1-8 X(04) pos 9-12 9(02) pos 13-14 9(02) pos 15-16	YY		
	BALANCE-TOTALS-DATA PRIOR-PERIODIC-DATE	X(33) S9(06) COMP pos 1-4	YYMMDD		OCCURS 4 TIMES
	UPDATE-TIME-STAMP UPDATE-DATE UPDATE-TIME	S9(12) COMP S9(06) COMP pos 1-4 S9(06) COMP pos 5-8	YYMMDD		
BAL-ADJUST-HDR					
	LIABILITY-DATE	S9(06) COMP	YYMMDD		
BAL-ADJUST-DTL					
	LIAB-DATE	S9(06) COMP	YYMMDD		redefine of sts-rate 9(2)V9(4) COMP
	EXP-DATE	S9(06) COMP	YYMMDD		redefine of transmitted-rate 9(2)V9(4) COMP
	UPDATE-TIME-STAMP UPDATE-DATE	x(8) S9(06) COMP pos 1-4	YYMMDD		
BAL-ADJ-RELEASE					
	ORIGINAL-LIABILITY-DATE	S9(06) COMP	YYMMDD		
	RELEASED-LIABILITY-DATE	S9(06) COMP	YYMMDD		
	UPDATE-TIME-	X(8)			

	STAMP UPDATE-DATE UPDATE-DATE	S9(06) COMP pos 1-4 S9(06) COMP pos 5-8	YYMMDD HHMMSS		
BAL-ADJ-DTL-HST					
	ORIGINAL-LIABILITY-DATE	S9(06) COMP	YYMMDD		
BAL-RLSE-NOTES					
	ORIGINAL-LIABILITY-DATE	S9(06) COMP	YYMMDD		
RECEIPTS-SUM					
	XPECT-XFER-DATE	S9(6) COMP	YYMMDD		
	QTR-PERIOD	9(4)	YYQQ		
	BANK-XFER-DATE	S9(6) COMP	YYMMDD		
	DATE-ADDED	S9(6) COMP	YYMMDD		
	RECEIPT-DATE	S9(6) COMP	YYMMDD		
	RECEIPT-POST-DATE	S9(6) COMP	YYMMDD		
	RETURN-DATE	S9(6) COMP	YYMMDD		
	RETURN-POST-DATE	S9(6) COMP	YYMMDD		
BANK-CONFIRM					
	BANK-DATE	S9(6) COMP	YYMMDD		
	DATE-ADDED	S9(6) COMP	YYMMDD		
	DATE-MATCHED	S9(6) COMP	YYMMDD		
BANK-LEDGER					
	LIABILITY-DATE	S9(6) COMP	YYMMDD		
	POST-DATE	S9(6) COMP	YYMMDD		

	CHECK-DATE	S9(6) COMP	YYMMDD		
	DATE-PAID	S9(6) COMP	YYMMDD		
	LAST-UPD-DATE	S9(6) COMP	YYMMDD		
CHECK-REGISTER					
	DUE-DATE	S9(6) COMP	YYMMDD		
	POST-DATE	S9(6) COMP	YYMMDD		
	CHECK-DATE	S9(6) COMP	YYMMDD		
	RECON-DATE	S9(6) COMP	YYMMDD		
	DEPOSIT-DATE	S9(6) COMP	YYMMDD		
CLIENT-LEDGER					
	TAX-ID FEE-INFO-1 FEE-COUNT-1 FEE-COUNT-2 FEE-PRICE FEE-FLAT-RATE	X(16). S9(16) COMP S9(6)V99 COMP pos 1-4 S9(6)V99 COMP pos 5-8 S9(6)V99 COMP pos 9-12 S9(6)V99 COMP pos 13-16			Redefine
	LIABILITY-DATE FUNDS-COLL-DATE	S9(6) COMP S9(6) COMP	YYMMDD YYMMDD		redefine
	POST-DATE	S9(6) COMP	YYMMDD		
	CHECK-DATE LIABILITY-PERIOD	S9(6) COMP S9(6) COMP	YYMMDD YYMMDD		redefine
	EXTRA-DATA CALC-DUE-DATE PRINT-DATE	X(6) 9(6) 9(6)	 YYMMDD YYMMDD		 redefine redefine
CHECK-REG-DTL					

	LIABILITY-DATE	S9(6) COMP	YYMMDD		
EFTPS-REF-DTL					
	DUE-DATE	S9(06) COMP	YYMMDD		
	ADD-DATE	S9(06) COMP	YYMMDD		
	CONFIRM-DATE	S9(06) COMP	YYMMDD		
SAMEDAY-XREF-DTL					
	ADD-DATE	S9(6) COMP	YYMMDD		
	CONFIRM-DATE	S9(6) COMP	YYMMDD		

⇒ STSQTR

FEE-PRICING-HDR					
	FEE-PRICING-ID	X(08) X(04) X(04)	 COST YYMM		ONLY IF COST AS FIRST FOUR AND POS 5 = 9
	DATE-ADDCHG	S9(6) COMP	YYMMDD		
	CREDIT-CATEGORY-DATA	PIC X(2)			OCCURS 3 TIMES
	CREDIT-CAT-APPLY-TO-CODES	PIC X(1)			OCCURS 4 TIMES
FEE-PRICING-DTL					
	FEE-PRICING-ID	X(08) X(04) X(04)	 COST YYMM		ONLY IF COST AS FIRST FOUR AND POS 5 = 9
	FEE-ID-AND-CODE. FEE-PRICING-KEY	 X(08) X(04) X(04)	 COST YYMM		ONLY IF COST AS FIRST FOUR AND POS 5 = 9

	TAX-CODE	X(04)			
CLIENT-FEE-HIST					
	COMBINE-PERIOD COMBINE-ID REM-CMBN-CODE RCC-CMPNY-CODE FILLER QTR-PERIOD	X(12) X(8) pos 1-8 X(8) X(4) pos 1-4 X(4) pos 5-8 X(4) pos 9-12	 YYQQ		 redefine
	CLIENT-PERIOD CLIENT-ID QTR-PERIOD-KEY	X(12) X(8) pos 1-8 X(4) pos 9-12	 YYQQ		
	APPLY-TO-CODE APPLY-TO-QTR	X(4) X(4)	 YYQQ		 redefine
	LIABILITY-PERIOD PMT-RCVD-DATE	S9(6) COMP S9(6) COMP	YYMMDD YYMMDD		 redefine
FEE-TRX-FILE					
	CLIENT-PERIOD CLIENT-ID QTR-PERIOD	X(12) X(8) pos 1-8 X(4) pos 9-12	 YYQQ		
	DATE-ADDCHG	S9(6) COMP	YYMMQQ		
	POST-DATE	S9(6) COMP	YYMMQQ		
	LIABILITY-DATE	S9(6) COMP	YYMMQQ		
QE-DATA					
	CLIENT-YEAR CLIENT-ID YEAR	X(10) X(08) pos 1-8 X(02) pos 9-10	 YY		
	CLT-TAX-ACCT-YR CLIENT-ID-KEY TAX-CODE	X(16) X(8) pos 1-8 X(4) pos 9-12			

	ACCOUNT-TYPE YEAR-KEY	9(2) pos 13-14 X(2) pos 15-16	 YY		
QE-DATA-HIST	CLT-TAX-ACCT-PER CLIENT-ID-KEY TAX-CODE ACCOUNT-TYPE QTR-PERIOD	X(18) X(8) pos 1-8 X(4) pos 9-12 9(2) pos 13-14 9(4) pos 15-18	 YYQQ		
	DATE-TIME	9(8) COMP			
QE-DATA-TRX-HDR					
	CLIENT-PERIOD CLIENT-ID QTR-PERIOD	X(12) X(8) pos 1-8 9(4) pos 9-12	 YYQQ		
	DATE-ADDCHG	9(6) COMP	YYMMDD		
	POST-DATE	S9(6) COMP	YYMMDD		
	AMEND-RPT-DATE	S9(6) COMP	YYMMDD		
QE-DATA-TRX-DTL					
	CLT-PERIOD-ENT CLIENT-ID QTR-PERIOD ENTRY-NO	x(14) X(8) pos 1-8 9(4) pos 9-12 9(2) pos 13-14	 YYQQ		
	CLIENT-PERIOD CLIENT-ID QTR-PERIOD	x(12) X(8) pos 1-8 9(4) pos 9-12	 YYQQ		
QE-FUNDS-XFER					
	DATE-ADDCHG	S9(6) COMP	YYMMDD		
	POST-DATE	S9(6) COMP	YYMMDD		
	LIABILITY-DATE	S9(6) COMP	YYMMDD		
	FUNDS-XFER-DATE	S9(6) COMP	YYMMDD		
QE-RECON					

	CLIENT-PERIOD CLIENT-ID QTR-PERIOD	x(12) X(8) pos 1-8 9(4) pos 9-12	 YYQQ		
	COMBINE-PERIOD COMBINE-ID QTR-PERIOD	X(12) X(8) pos 1-8 X(4) pos 9-12	 YYQQ		
	FUNDS-XFER-DATE	S9(6) COMP	YYMMDD		
	RPT-RECEIPT-DATE	S9(6) COMP	YYMMDD		
	COMPLETE-DATE	S9(6) COMP	YYMMDD		
QE-RECON-NOTES					
	CLIENT-PERIOD CLIENT-ID QTR-PERIOD	X(12) X(8) pos 1-8 9(4) pos 9-12	 YYQQ		
QE-TAX-ID-XREF					
	CLIENT-PERIOD CLIENT-ID QTR-PERIOD	X(12) X(8) pos 1-8 9(4) pos 9-12	 YYQQ		
	TAX-CODE-ID-PER CONSOL-TAX-CODE TAX-ID QTR-PERIOD-KEY	X(24) X(4) pos 1-4 X(16) pos 5-20 9(4) pos 21-24	 YYQQ		
TAX-SUMMARY-HIST					
	COMBINE-PERIOD COMBINE-ID QTR-PERIOD	X(12) X(8) pos 1-8 X(4) pos 9-12	 YYQQ		
	CLIENT-PERIOD CLIENT-ID QTR-PERIOD	X(12) X(8) pos 1-8 9(4) pos 9-12	 YYQQ		
TAX-RETURN-SUM					
	TAX-CODE-ID-PER CONSOL-TAX-CODE	X(24) X(4) pos 1-4			

	TAX-ID QTR-PERIOD-KEY	X(16) pos 5-20 9(4) pos 21-24	 YYQQ		
TAX-RETURN-DTL					
	TAX-CODE-ID-YR CONSOL-TAX-CODE TAX-ID YEAR	X(22) X(4) pos 1-4 X(16) pos 5-20 X(2) pos 21-22	 YY		
	CLT-TAX-ACCT-YR CLIENT-ID TAX-CODE ACCOUNT-TYPE YEAR-KEY	X(16) X(8) pos 1-8 X(4) pos 9-12 9(2) pos 13-14 X(2) pos 15-16	 YY		
PROCESS-LOG					
	QTR-PERIOD	9(4)	YYQQ		
	DATE-ADDED	S9(6)	YYMMDD		
TAX-YE-SUMMARY					
	TAX-CODE-ID-YR CONSOL-TAX-CODE TAX-ID YEAR	X(22) X(4) pos 1-4 X(16) pos 5-20 X(2) pos 21-22	 YY		
	TAX-CODE-ID-YR-2 CONSOL-TAX-CODE TAX-ID YEAR	X(22) X(4) pos 1-4 X(16) pos 5-20 X(2) pos 21-22	 YY		
TAX-DLY-SUMMARY					
	TAX-CODE-ID-PER CONSOL-TAX-CODE TAX-ID QTR-PERIOD-KEY	X(24) X(4) pos 1-4 X(16) pos 5-20 9(4) pos 21-24	 YYQQ		
	DATE-ADDCHG	S9(6) COMP	YYMMDD		

TAX-DLY-DEPOSIT					
	TAX-CODE-ID-PER CONSOL-TAX-CODE TAX-ID QTR-PERIOD-KEY	X(24) X(4) pos 1-4 X(16) pos 5-20 9(4) pos 21-24	 YYQQ		
	DATE-ADDCHG	S9(6) COMP	YYMMDD		
	LIAB-DATES	S9(06) COMP	YYMMDD		array 32 times
	CHECK-DATES	S9(06) COMP	YYMMDD		array 32 times
QTR-TAX-HDR					
	QTR-PER	X(04)	YYQQ		
QTR-TAX-DTL					
	QTR-PER	X(04)	YYQQ		

⇒ STSCTL

CONTROL				**QSCTLFIL.COPYLIB** **TFCTLFIL.COPYLIB**	depending on the key, date fields could hold valid dates or invalid information
	QTR-PERIOD	9(4)	YYQQ	CTL-QTR-PERIOD	
	FUNDS-XFER-DATE	S9(6) COMP	YYMMDD	CTL-FUNDS-XFER-DATE	
	CURR-PERIOD	9(5) COMP	Yynnn	CTL-CURR-PERIOD	
	IRS-TAPE-APPL-DATE	9(6) COMP	YYMMDD	CTL-IRS-TAPE-APPL-DATE	
	QE-LATE-DATE-INFO	S9(6) COMP	YYMMDD	CTL-QE-LATE-DATE-INFO	
	QE-FUNDS-XFER-DAY-INFO	S9(6) COMP	YYMMDD	CTL-QE-FUNDS-XFER-DAY-INFO	
	CHECK-TRX-POST-	S9(6) COMP	YYMMDD	CTL-CHECK-TRX-POST-	

	DATE			DATE	
GROUP-MASTER				**QSGMRFIL.COPYLIB / TFSCMFIL.COPYLIB**	
	GROUP-MASTER-RECORD GROUP-ID GROUP-DESCRIPTION GROUP-SECURITY-LVL GROUP-CLIENT-CODE GROUP-CLIENT-DIVISION GROUP-COMPANY-CODE GROUP-PROC-ASSOC-ID GROUP-STS-BANK-ID GROUP-PR-PROC-ID GROUP-PR-COLL-ID GROUP-CHECK-FIELD GROUP-DEPT-CODE FILLER FILLER GROUP-MASTER-RECORD-ZERO FILLER GROUP-ZERO-CHECK-DATE GROUP-ZERO-ACCT-PASSWORD FILLER	 9(6). X(20). 9(2) COMP X(5). X(3). X(4). X(4). X(2) X6) X(6). X(2) . X(2). X(5). X(7) 9(6). 9(6)		 GROUP-ID GROUP-DESCRIPTION GROUP-SECURITY-LVL GROUP-CLIENT-CODE GROUP-CLIENT-DIVISION GROUP-COMPANY-CODE GROUP-PROC-ASSOC-ID GROUP-STS-BANK-ID GROUP-PR-PROC-ID GROUP-PR-COLL-ID GROUP-CHECK-FIELD GROUP-DEPT-CODE GROUP-ZERO-CHECK-DATE	 redefine no records found

		X(8). X(47)		GROUP-ZERO-ACCT-PASSWORD	with dates, assume YYMMDD
GROUP-DETAIL				**QSGDLFIL.COPYLIB**	
REQUEST-TRX				**QSRQTFIL.COPYLIB / TFRQTFIL.COPYLIB**	
	DATE-ADDED	S9(6) COMP	YYMMDD	RQT-DATE-ADDED	
	DATE-PROCESSED	S9(6) COMP	YYMMDD	RQT-DATE-PROCESSED	
	QTR-PERIOD	9(4)	YYQQ	RQT-QTR-PERIOD	
	STARTING-LIAB-DATE	9(6)	YYMMDD	RQT-STARTING-LIAB-DATE	
	ENDING-LIAB-DATE	9(6)	YYMMDD	RQT-ENDING-LIAB-DATE	
	STARTING-POST-DATE	9(6)	YYMMDD	RQT-STARTING-POST-DATE	
	ENDING-POST-DATE	9(6)	YYMMDD	RQT-ENDING-POST-DATE	
	SELECT-DATA TAX-CODE-ID-PERIOD TAX-CODE-AND-ID TAX-ID RETURN-QTR-PERIOD	X(80) X(24) pos 1-24 X(4) pos 1-4 X(16) pos 5-20 9(4) pos 21-24	 YYQQ	RQT-SELECT-DATA RQT-TAX-CODE-ID-PERIOD RQT-TAX-CODE-AND-ID RQT-RETURN-QTR-PERIOD	
TEXT-DETAIL				**QSTXTFIL.COPYLIB / TFTXTFIL.COPYLIB**	
	CLIENT-PERIOD CLIENT-ID-KEY QTR-PERIOD-KEY	X(12) X(8) pos 1-8 X(4) pos 9-12	 YYQQ	TXT-CLIENT-PERIOD TXT-CLIENT-ID-KEY XT-QTR-PERIOD-KEY	
	TEXT-KEY CLIENT-ID QTR-PERIOD TEXT-TYPE	X(14) X(8) pos 1-8 X(4) pos 9-12 X(2) pos 13-14	 YYQQ	TXT-TEXT-KEY TXT-CLIENT-ID TXT-QTR-PERIOD TXT-TEXT-TYPE	
	TXT-TEXT	X(70)			

	FILLER CLIENT-NOTE-TEXT CLIENT-NOTE-USER CLIENT-NOTE-DATE FILLER	X(70) X(50) pos 1-50 X(8) pos 51-58 9(6) pos 59-64 X(6) pos 65-70	YYMMDD		redefine
	PHS-LAST-UPD-DATE	S9(6) COMP	YYMMDD		
	TXT-TEXT-KEY TXT-QTR-PERIOD TXT-FILLER TXT-TEXT-TYPE	X(14) X(04) pos 1-4 X(08) pos 5-12 X(02) pos 13-14	YYQQ		
	LAST-UPD-DATE	S9(06) COMP	YYMMDD	TCD-LAST-UPD-DATE	
RJE-FILE-HEADER				**TFRFHFIL.COPYLIB**	
	LAST-UPDATE-DATE	S9(6) COMP	YYMMDD	RFH-LAST-UPDATE-DATE	
	DATE-RECEIVED	S9(6) COMP	YYMMDD	RFH-DATE-RECEIVED	
	DATE-RERUN	S9(6) COMP	YYMMDD	RFH-DATE-RECEIVED	
	DATE-POSTED	S9(6) COMP	YYMMDD	RFH-DATE-POSTED	
BANK-FILE				**QSBNKFIL.COPYLIB / TFBNKFIL.COPYLIB**	
	START-DATE	S9(6) COMP	YYMMDD	BANK-START-DATE	
	END-DATE	S9(6) COMP	YYMMDD	BANK-END-DATE	
	LAST-UPD-DATE	S9(6) COMP	YYMMDD	BANK-LAST-UPD-DATE	
	TAPE-APPROVAL-DATE	S9(6) COMP	YYMMDD	BANK-TAPE-APPROVAL-DATE	
BANK-BALANCE				**TFBKBFIL.COPYLIB**	
	STMNT-DATE	S9(6) COMP	YYMMDD	BANK-STMNT-DATE	
	STMNT-UPD-DATE	S9(6) COMP	YYMMDD	BANK-STMNT-UPD-DATE	
	CURR-DATE	S9(6) COMP	YYMMDD	BANK-CURR-DATE	
	CURR-UPD-DATE	S9(6) COMP	YYMMDD	BANK-CURR-UPD-DATE	
AUDIT-FILE					
	AUDIT-DATE	S9(6) COMP	YYMMDD		

RJE-YTD-CURR					
	LIABILITY-DATE	9(6) COMP	YYMMDD		
	LAST-UPD-DATE	9(6) COMP	YYMMDD		
RJE-YTD-QTR					
	QTR-PERIOD-1	9(4)	YYQQ		
	QTR-PERIOD-2	9(4)	YYQQ		
	QTR-PERIOD-3	9(4)	YYQQ		
	QTR-PERIOD-4	9(4)	YYQQ		
	LAST-UPD-DATE	9(6) COMP	YYMMDD		
TI-DATA					
	TI-DATE	S9(06) COMP	YYMMDD		
	EFF-DATE	S9(06) COMP	YYMMDD		
	EFF-QUARTER	X(04)	YYQQ		
	CREATE-DATE	S9(06) COMP	YYMMDD		
	LAST-UPD-DATE	S9(06) COMP	YYMMDD		
AUDIT-DETAIL					
	RPTED-DATE	S9(06) COMP	YYMMDD		
	TI-DATE	S9(06) COMP	YYMMDD		
	EFFECTIVE-DATE	S9(06) COMP	YYMMDD		
	LAST-UPD-DATE	S9(06) COMP	YYMMDD		
	RATE-EFFECTIVE-QTR	X(4)	YYQQ		
BILLING-CONTROL					
	LAST-PERIOD-START	S9(06) COMP	YYMMDD		
	LAST-PERIOD-END	S9(06) COMP	YYMMDD		
	CURR-PERIOD-START	S9(06) COMP	YYMMDD		
	CURR-PERIOD-END	S9(06) COMP	YYMMDD		

NEXT-TRX-NUMBERS					
	LAST-UPDATE-DATE	S9(6)COMP	YYMMDD		
EFTPS-CONTROL					
	813-JULIAN-DATE	X(06)	YYnnn		
	CONTROL-UPD-DATE	S9(06) COMP	YYMMDD		
EFTPS-REJECT					
	REJECT-UPD-DATE	S9(06) COMP	YYMMDD		

⇒ STSINA

INA-LEDGER-01 - 08				**TFILDFIL.COPYLIB**	all ina-ledger sets look the same
	TAX-ID FEE-INFO-1 FEE-COUNT-1 FEE-COUNT-2 FEE-PRICE FEE-FLAT-RATE	X(16). X(16). S9(6)V99 COMP pos 1-4 S9(6)V99 COMP pos 5-8 S9(6)V99 COMP pos 9-12 S9(6)V99 COMP pos 13-16			redefine; ISSUE; require more ana. To see when amounts are used over tax-id
	LIABILITY-DATE FUNDS-COLL-DATE	S9(6) COMP S9(6) COMP	YYMMDD YYMMDD		redefine
	POST-DATE	S9(6) COMP	YYMMDD		
	CHECK-DATE LIABILITY-PERIOD	S9(6) COMP S9(6) COMP	YYMMDD YYMMDD		redefine
	EXTRA-DATA CALC-DUE-DATE PRINT-DATE	X(6) 9(6) 9(6)	 YYMMDD YYMMDD		 redefine redefine
CES-HISTORY					
	TAX-ID FEE-INFO-1 FEE-COUNT-1 FEE-COUNT-2	X(16). X(16). S9(6)V99 COMP pos 1-4 S9(6)V99 COMP pos 5-8			redefine

	FEE-PRICE FEE-FLAT-RATE	S9(6)V99 COMP pos 9-12 S9(6)V99 COMP pos 13-16			
	LIABILITY-DATE FUNDS-COLL-DATE	S9(6) COMP S9(6) COMP	YYMMDD YYMMDD		 redefine
	POST-DATE	S9(6) COMP	YYMMDD		
	CHECK-DATE LIABILITY-PERIOD	S9(6) COMP S9(6) COMP	YYMMDD YYMMDD		 redefine
	EXTRA-DATA CALC-DUE-DATE PRINT-DATE	X(6) 9(6) 9(6)	 YYMMDD YYMMDD		 redefine redefine
INA-SUMMARY					
	KEY CLT-STATUS QTR-PERIOD CLIENT-ID ACTUAL-PERIOD	X(18) X(2) pos 1-2 9(4) pos 3-6 X(8) pos 7-14 9(4) pos 15-18			

⇒ STSCIS

CIS-TABS				**QSCDEFIL.COPYLIB**	depending on key, filler92 can be redefined any numerous ways
	FILLER92 DESCRIPTION RESPON-DEPT OPEN-DATE CLOSE-DATE SURETY-LEVEL BILLABLE REASON-FEES	X(92) X(30) pos 1-30 X(02) pos 31-32 9(06) pos 33-38 9(06) pos 39-44 9(03) pos 45-47 X(02) pos 48-49 9(06) pos 50-55	 YYMMDD YYMMDD		
CIS-TABS	FILLER92	X(92)		**QSCTBFIL.COPYLIB**	
CIS-TABS	FILLER92	X(92)		**QSRFDFIL.COPYLIB**	

	REQ-TYPE-FLAG CHANGE-TYPE-FLAG	X(02) pos 1-2 X(02) pos 3-4			
CIS-TABS	FILLER92	X(92)		**QSSPCFIL.COPYLIB**	
CIS-TABS	FILLER92 NUM-EQUIV NO-TIMES NO-ACTIVE CLOSE-OPEN-STEP CLOSE-TYPE ADD-FLAG	X(92) S9(03) pos 1-3 S9(03) pos 4-6 S9(03) pos 7-10 X(02) pos 11-12 X(02) pos 13-14 X(02) pos 15-16		**QSSPDFIL.COPYLIB**	
CIS-TABS	FILLER92 CREATE-WIP INITIATOR RECEIVER CLOSE-TO-STEP INITIATE-STEP REQ-FIELD-TAB	X(92) X(02)pos 1-2 X(04)pos 3-6 X(04) pos 7-10 X(02) pos 11-12 X(06) pos 13-18 X(06) pos 19-24		**QSSPTFIL.COPYLIB**	
CIS-TABS	FILLER92 SCOPE NAME-TYPE SUPR-IN AE-XREF	X(92) X(04) pos 1-4 X(08) pos 5-12 X(03) pos 13-15 X(03) pos 16-18		**QSUSTFIL.COPYLIB**	
CIS-TABS	FILLER92 USER	X(92) X(04) pos 1-4		**QSUTTFIL.COPYLIB**	
INQ-DTL				**QSINQFIL.COPYLIB / TFINQFIL.COPYLIB**	
	PERIOD	X(4)	YYQQ		
	NOTICE-DATE	9(6)	YYMMDD		
	LOGIN-DATE	9(6)	YYMMDD		
	LOGOFF-DATE	9(6)	YYMMDD		
	MAIL-DATE	9(6)	YYMMDD		
	LOGIN-DATE-SORT	9(6)	YYMMDD		
	START-STEP-TS	9(12)	YYMMDDH		

			HMMSS		
	TOT-AMOUNT-CIS				I3 invalid integer type
	TAX-AMOUNT-CIS				I3 invalid integer type
	PEN-AMOUNT				I3 invalid integer type
	INT-AMOUNT				I3 invalid integer type
	FORM-AMOUNT				I3 invalid integer type
NOTES-DTL				**QSNOTFIL.COPYLIB TFNOTFIL.COPYLIB**	
	CREATE-TS	9(12)	YYMMDDH HMMSS	NOTES-DTL-CREATE-TS	
STEPS-DTL				**QSSTPFIL.COPYLIB**	
	START-STEP-TS	9(12)	YYMMDDH HMMSS		
	STOP-STEP-TS	9(12)	YYMMDDH HMMSS		
TAX-ALERT-DTL				**QSTADFIL.COPYLIB**	
	ALERT-DATE	9(6)	YYMMDD		

⇒ TAXMAN

KEY-MASTER					
	DATE	X(6)	YYMMDD		

⇒ LEDGER

MONTHLY-LEDGER					

	MONTHLY-PERIOD	X(4)	YYQQ		
PROCESS-INFO					
	LIAB-DATE	S9(6) COMP	YYMMDD		
	POST-DATE	S9(6) COMP	YYMMDD		

⇒ FAX

FAX-DETAIL					
	ENTRY-NO	9(02)			contains binary numbers
	TRANS-DATE	S9(06) COMP	YYMMQQ		
	CHECK-DATE	S9(06) COMP	YYMMQQ		
FAX-NOTES-HDR					
	DATE-TIME-SENT	S9(12)	YYMMDDH HMMSS		
	LIABILITY-DATE	S9(06) COMP	YYMMDD		
	UPDATE-TIME-STAMP	S9(12) COMP	YYMMDDH HMMSS		

⇒ RDATA

RETURN-DATA					
	RDATA1	X(232)			holds forms information
	RDATA2	X(232)			holds forms information
FIELDNAME-KEY					
	LAST-UPD-DATE	S9(6) COMP	YYMMDD		

⇒ WGE

WAGE-HEADER					
	CLIENT-QTR-STATE CLIENT-ID PERIOD STATE	X(14) X(08) pos 1-8 X(04) pos 9-12 X(02) pos 13-14	 YYQQ		
	STATE-QTR-EIN STATE PERIOD EIN	x(22) X(02) pos 1-2 X(04) pos 3-6 X(16) pos 7-22	 YYQQ		
	DATE-ADDCHG	S9(06) COMP	YYMMDD		
	POST-DATE	S9(06) COMP	YYMMDD		
WAGE-DETAIL					
	NEW-HIRE-DATE	S9(07)V99 COMP	**MMDDYY**		
LCL-WAGE-HEADER					
	CLIENT-QTR-LOCAL CLIENT-ID PERIOD STATE LOCAL	x(16) X(08) pos 1-8 X(04) pos 9-12 X(02) pos 13-14 X(02) pos 15-16	 YYQQ		
	LOCAL-QTR-EIN STATE LOCAL PERIOD EIN	x(24) X(02) pos 1-2 X(02) pos 3-4 X(04) pos 5-8 X(16) pos 9-24	 YYQQ		
	DATE-ADDCHG	S9(06) COMP	YYMMDD		
	POST-DATE	S9(06) COMP	YYMMDD		
CLIENT-PERIODS					
	CLIENT-PERIOD CLIENT-ID PERIOD	X(12) X(08) pos1-8 X(04) pos 9-12	 YYQQ		

	CLIENT-ACT-START	S9(06) COMP	YYMMDD		
	FINAL-WAGE-DATE	S9(06) COMP	YYMMDD		
WAGE-HEADER-A					
	CLIENT-QTR-STATE CLIENT-ID PERIOD STATE	X(14) X(08) pos 1-8 X(04) pos 9-12 X(02) pos 13-14	YYQQ		
	STATE-QTR-EIN STATE PERIOD EIN	x(22) X(02) pos 1-2 X(04) pos 3-6 X(16) pos 7-22	YYQQ		
	CLIENT-PERIOD CLIENT-ID PERIOD	X(12) X(08) pos 1-8 X(04) pos 9-12	YYQQ		
	ENTRY-NO	X(02)			may hold binary numbers
	DATE-ADDCHG	S9(06) COMP	YYMMDD		
	POST-DATE	S9(06) COMP	YYMMDD		
WAGE-DETAIL-A					
	CLIENT-PERIOD CLIENT-ID PERIOD	X(12) X(08) pos 1-8 X(04) pos 9-12	YYQQ		
	CLIENT-QTR-STATE CLIENT-ID PERIOD STATE	X(14) X(08) pos 1-8 X(04) pos 9-12 X(02) pos 13-14	YYQQ		
	ENTRY-NO	X(02)			but may hold binary numbers
	NEW-HIRE-DATE	S9(07)V99 COMP	**MMDDYY**		
LCL-WAGE-HDR-A					
	CLIENT-QTR-LOCAL CLIENT-ID	x(16) X(08) pos 1-8			

	PERIOD STATE LOCAL	X(04) pos 9-12 X(02) pos 13-14 X(02) pos 15-16	YYQQ		
	LOCAL-QTR-EIN STATE LOCAL PERIOD EIN	x(24) X(02) pos 1-2 X(02) pos 3-4 X(04) pos 5-8 X(16) pos 9-24	YYQQ		
	CLIENT-PERIOD CLIENT-ID PERIOD	X(12) X(08) pos 1-8 X(04) pos 9-12	YYQQ		
	ENTRY-NO	X(02)			may hold binary
	DATE-ADDCHG	S9(06) COMP	YYMMDD		
	POST-DATE	S9(06) COMP	YYMMDD		
LCL-WAGE-DTL-A					
	CLIENT-PERIOD CLIENT-ID PERIOD	X(12) X(08) pos 1-8 X(04) pos 9-12	YYQQ		
	CLIENT-QTR-LOCAL CLIENT-ID PERIOD STATE LOCAL	x(16) X(08) pos 1-8 X(04) pos 9-12 X(02) pos 13-14 X(02) pos 15-16	YYQQ		
	ENTRY-NO	X(02)			may hold binary
WC-SUMMARY					
	STATE-QTR-EIN STATE PERIOD EIN	x(22) X(02) pos 1-2 X(04) pos 3-6 X(16) pos 7-22	YYQQ		

⇒ STSW2F

W2-TAX-EIN-MAST					
	REPORT-YEAR REPORT-CEN REPORT-YR	9(04) 9(002) pos 1-2 9(002) pos 3-4	CCYY CC YY		
	STATUS-DATE	9(009) COMP	YYMMDD		
	LAST-UPD-DATE	9(009) COMP	YYMMDD		
W2-DATA					
	EMPE-YEAR	X(04)	CCYY		
	STATUS-DATE	S9(6) COMP	YYMMDD		
	LAST-UPD-DATE	S9(6) COMP	YYMMDD		
W2C-DATA					
	W2C-KEY RPTING-YEAR EMPLR-FEIN EMPLE-SSN RECORD-TYPE	x(26) X(004) pos 1-4 X(012) pos 5-16 X(010) pos 17-26 X(002) pos 27-28	CCYY		

Table 2:

HANDLING OF INVALID DATA.

```
SQL> desc payee_record
 Name          Null?  Type
 ------------------------------- -------- ----
 PAYEE_TABLE           VARCHAR2(6)
 PAYEE_ID            VARCHAR2(14)
 PAYEE_NAME            VARCHAR2(30)
 PAYEE_ADDR1           VARCHAR2(30)
 PAYEE_ADDR2           VARCHAR2(30)
 PAYEE_CITY           VARCHAR2(16)
 PAYEE_STATE           VARCHAR2(2)
 PAYEE_ZIP           VARCHAR2(10)
 PAYEE_ESCROW_FLAG         VARCHAR2(1)
 PAYEE_DAILY_BANK_ID        VARCHAR2(2)
 PAYEE_QE_BANK_ID          VARCHAR2(2)
 PAYEE_PI_BANK_ID          VARCHAR2(2)
 PAYEE_EFT_OK_FLAG         VARCHAR2(1)
 PAYEE_ABA_TRA_NO          VARCHAR2(8)
 PAYEE_TRA_CHECK_DIG        VARCHAR2(1)
 PAYEE_ACCT_NO          VARCHAR2(17)
 PAYEE_TXP_TAX_TYPE         VARCHAR2(5)
 PAYEE_PERIOD_END_FREQ       VARCHAR2(1)
 PAYEE_PERIOD_END_EOP_DUE      VARCHAR2(1)
 PAYEE_TXP_AMT_TYPE_1        VARCHAR2(1)
 PAYEE_TXP_TAX_CODE_1        VARCHAR2(4)
 PAYEE_TXP_AMT_TYPE_2        VARCHAR2(1)
 PAYEE_TXP_TAX_CODE_2        VARCHAR2(4)
 PAYEE_TXP_AMT_TYPE_3        VARCHAR2(1)
```

```
PAYEE_TXP_TAX_CODE_3        VARCHAR2(4)
PAYEE_TAX_ID_LEN            NUMBER(2)
PAYEE_ID_NUMERIC_FLAG       VARCHAR2(1)
PAYEE_ID_ZERO_FILL_FLAG     VARCHAR2(1)
PAYEE_ACCESS_CODE_LEN_X     VARCHAR2(2)
PAYEE_ADDENDA_TAX_ID_FMT    VARCHAR2(16)
PAYEE_UPD_DATE              DATE
PAYEE_UPD_USER              VARCHAR2(4)
PAYEE_ZERO_ACH_FLAG         VARCHAR2(1)
FILLER_0                    VARCHAR2(3)

SQL>
```

Table 3:

BRIDGEWARE SCRIPT USED TO HANDLE INVALID DATA.

```
Entity
 Entity-key  x8
 Entity-data  x140

Entity-data



Script used;


Open cprx image ststax pass={password} mode=1

Open oradb remote user={userid} pass={password} &
  Oracle {dblogin}/{dbpass} &
Home =/opt/oracle/app/oracle/product/8.1.7

Set oradb commitrate 10000

format payee_record              :     record
 PAYEE_TABLE       :   x6
 PAYEE_ID        :   x14
 PAYEE_NAME        :   x30
 PAYEE_ADDR1       :   x30
 PAYEE_ADDR2       :   x30
```

```
 PAYEE_CITY        :  x16
 PAYEE_STATE       :  x2
 PAYEE_ZIP         :  x10
 PAYEE_ESCROW_FLAG     :  x1
 PAYEE_DAILY_BANK_ID     :  x2
 PAYEE_QE_BANK_ID      :  x2
 PAYEE_PI_BANK_ID      :  x2
 PAYEE_EFT_OK_FLAG     :  x1
 PAYEE_ABA_TRA_NO      :  x8
 PAYEE_TRA_CHECK_DIG     :  x1
 PAYEE_ACCT_NO       :  x17
 PAYEE_TXP_TAX_TYPE      :  x5
 PAYEE_PERIOD_END_FREQ     :  x1
 PAYEE_PERIOD_END_EOP_DUE    :  x1
 PAYEE_TXP_AMT_TYPE_1      :  x1
 PAYEE_TXP_TAX_CODE_1      :  x4
 PAYEE_TXP_AMT_TYPE_2      :  x1
 PAYEE_TXP_TAX_CODE_2      :  x4
 PAYEE_TXP_AMT_TYPE_3      :  x1
 PAYEE_TXP_TAX_CODE_3      :  x4
 PAYEE_TAX_ID_LEN      :  x2
 PAYEE_ID_NUMERIC_FLAG     :  x1
 PAYEE_ID_ZERO_FILL_FLAG    :  x1
 PAYEE_ACCESS_CODE_LEN_X    :  x2
 PAYEE_ADDENDA_TAX_ID_FMT    :  x16
 PAYEE_UPD_DATE       :  z6
 PAYEE_UPD_USER       :  x4
 PAYEE_ZERO_ACH_FLAG     :  x1
 FILLER_0         :  x3
end
```

```
define  entity      using cprx.entity

Read payee_in = cprx.entity format payee_record &
  for cprx.entity_key='PAYEE'

Setvar cprx.payee_table = 'PAYEE'
Setvar cprx..payee_id = payee-id
Setvar cprx.payee_name = payee-name

If last-upd-date = 991231
  Setvar cprx.last_upd_date = 30001231
Else if last-upd-date > 0
  Setvar cprx.last_upd_date = last-upd-date
Else
  Setvar cprx.last_upd_date = null
Endif



Copy cprx to payee_table
endread
```