



TIPS
TurboImage Port to SQL

User and Programmers Manual

Version: 1.0.9
1/31/2012

I. Introduction	3
How TIPS works.....	3
II. RUNNING TIPS	4
TIPS Input.....	4
TIPS Output	4
TIPS Run Example	5
Output Files and Error Files.....	6
Steps to Build and Populate a Database.....	7
III. CONFIGURATION (CONFIG) FILE.....	9
Commands	9
IV. USER EXITS and TIPS RUNTIME LIBRARY	16
User Exits.....	16
TIPSUEInit	17
TIPSUEConvertTable.....	18
TIPSUEConvertRecord.....	21
Runtime Library.....	23
TIPSGetFieldAttributes	23
TIPSPutFieldAttributes.....	25
TIPSInsertField.....	26
TIPSRemoveField.....	27
TIPSGetFieldValue.....	28
TIPSPutFieldValue	28
TIPSSortFieldsByName.....	29
TIPSExplainError	29
V. HELPFUL HINTS	30
Sample Files.....	30
Conversion Errors	30
Rerunning TIPS	30
SQL SERVER TIPS	31
UNIX / ORACLE TIPS	31
VI. Sample CONFIG FILE.....	33

I. Introduction

TIPS is a tool which converts HP3000 TurboImage databases, KSAM files and flat files to a relational database on Windows or Unix. TIPS allows you to do database structure or field content changes during the extraction process. It is a generic tool which can be used to migrate any HP3000 TurboImage database, and also any KSAM or flat files which are defined in an HP3000 POWERHOUSE schemas.

How TIPS works

1. You run TIPS. It asks you for a database name and password, and/or a POWERHOUSE schema name, and the kind of SQL DBMS you're transferring to (Microsoft SQL Server, ORACLE or MySQL)
2. TIPS examines the IMAGE database structure, and parses the POWERHOUSE file, and builds a script to create the SQL schema on the target machine.
3. TIPS then extracts the HP3000 data into a series of delimited files, one per dataset or KSAM/flat file. At the same time, it builds the load scripts necessary for loading this data to SQL Server, Oracle or MySQL databases.
4. While it's building the SQL schema and extracting data, TIPS is invoking user exits. These user exits provide you the opportunity to modify the database structure, or the contents of fields, as you wish. The TIPS runtime procedures can from within the the user exits, to do the following:
 - GetFieldAttributes: Look at the current definition of a field (name, length, type, etc)
 - PutFieldAttributes: Modify the current definition of a field
 - InsertField, RemoveField: Add or Remove a field
 - GetFieldValue, PutFieldValue: Look at, or modify the contents of a field
5. You then transfer the SQL creation script, the extract files and the load scripts to your target machine build the database using the SQL creation script, and populate the database using the extract files and load scripts.

TIPS comes with user exit examples to help you with the complex data transformations you might want to do, like splitting a single record into multiple records, or splitting a single table into multiple tables.

TIPS also does lots of common activities without the use of user exit code. These include date conversions, renaming fields or tables, specifying implied decimals for fields, specifying number of decimal places of precision for real numbers, removing fields or tables, re-typing fields, etc. These activities are controlled by a configuration file you can modify (for example, date conversion rules, implied decimals for specific fields, renaming fields which conflict with SQL reserved words, etc.).

To get started, you might want to just run TIPS and point it at a database to see what it does. After that, you can modify your CONFIG file and/or add user exits to handle your specific data transformation needs.

Please call us at 408-293-7500 with any questions, concerns, or just to help you get started.

II. RUNNING TIPS

To execute TIPS, you will generally be logged into the account you're extracting data from. The TIPS software should be installed on your system in the account COMP3.

The command to run TIPS is:

```
RUN TIPS.PUB.COMP3;XL=""TIPSXL.PUB.COMP3"
```

If you have user exits (described in chapter 4), the command would be:

```
RUN TIPS.PUB.COMP3;XL=<your user exit XL names>,TIPSXL.PUB.COMP3"
```

WARNING: BE SURE TO STATE THE FULL FILENAME, WITH THE GROUP AND ACCOUNT, IN YOUR USER EXIT XL NAME. OTHERWISE YOUR USER EXITS WILL NOT BE CALLED!!!

TIPS Input

Besides the files you are transforming from (database, powerhouse schema, flat files, KSAM files) TIPS has one additional input file, named CONFIG. You use the CONFIG file to specify various configuration options for your particular migration operation, like date conversion logic, number of implied decimals, sets and items to exclude or rename, etc. See chapter 3, CONFIG file, for a complete description. If you don't have a local CONFIG file, you can issue the file equation

```
FILE CONFIG=CONFIG.PUB.COMP3
```

before running the program. This is an empty CONFIG file provided with TIPS.

TIPS Output

TIPS produces the following output files:

- SCHSQL – text file for creating schema on target system.
- ef0001, ef0002, etc.: For extracted dataset or file, one delimited data extract file, starting with ef0001, then ef0002, etc.
- (Oracle only): lf0001, lf0002, etc.: If you specify Oracle as the target database, TIPS will create one lf file for each EF file. The lf file is used in conjunction with the ef file to load data to Oracle. lf files describe to Oracle what fields are being loaded, and their format.
- (SQL Server only and MySQL): TABLOADS. This is a single text file which is used by SQL Server and MySQL to load the ef#### data files into the db. Similar to the lf files described above for Oracle; except instead of one per extracted file (like the lf files), it's just one single TABLOADS file to load all the ef files.

TIPS Run Example

The following is an example of the TIPS runtime dialogue:

```
:RUN TIPS.PUB.COMP3;XL="TIPSL.PUB.COMP3"

TIPS VERSION 1.0.10
COPYRIGHT 2004-2012 COMP THREE INC. ALL RIGHTS RESERVED
PHONE: 408-293-7500 FAX: 408-293-7501
EMAIL: inquiries@compthree.com WEB: www.compthree.com
INPUT TYPE OF DBMS FOR OUTPUT SQL CODE:
1=MICROSOFT SQL SERVER
2=ORACLE
3=MYSQL
(Q TO QUIT PROGRAM)
1

INPUT FIELD DELIMITER CHARACTER FOR EXTRACT FILES
(%xx for unprintable character, where xx is a decimal number)%9

INPUT NAME OF POWERHOUSE SCHEMA FILE FOR KSAM AND FLAT FILES: <return>
OK, NO KSAM OR FLAT FILES WILL BE EXTRACTED.

INPUT NAME OF DATABASE TO OPEN: SALESDB
INPUT READ PASSWORD: ;
WRITING SQL TABLE CREATION SCRIPT...
EXTRACTING NAME-MAST          INTO ef0001; FOR LOAD TO name_mast
EXTRACTING DODAAD-MAST        INTO ef0002; FOR LOAD TO dodaad_mast
EXTRACTING DODAAD-MAST1       INTO ef0003; FOR LOAD TO dodaad_mast1
EXTRACTING INT-RATE-MAST      INTO ef0004; FOR LOAD TO int_rate_mast
EXTRACTING NBU-MAST           INTO ef0005; FOR LOAD TO nbu_mast
EXTRACTING SHIP-LOG-DETL      INTO ef0006; FOR LOAD TO ship_log_detl
EXTRACTING DD250-DETL         INTO ef0007; FOR LOAD TO dd250_detl
EXTRACTING DD250-AUX-DETL     INTO ef0008; FOR LOAD TO dd250_aux_detl
EXTRACTING HAC-PACK-DETL      INTO ef0009; FOR LOAD TO hac_pack_detl
EXTRACTING SHIP-ITEMS-DETL    INTO ef0010; FOR LOAD TO ship_items_detl
EXTRACTING SERIAL-NO-DETL     INTO ef0011; FOR LOAD TO serial_no_detl
EXTRACTING INV-HIST-DETL      INTO ef0012; FOR LOAD TO inv_hist_detl
EXTRACTING INV-DESCR-DETL     INTO ef0013; FOR LOAD TO inv_descr_detl
EXTRACTING INV-COM-DETL       INTO ef0014; FOR LOAD TO inv_com_detl
```

```
EXTRACTING PAY-HIST-DETL      INTO ef0015; FOR LOAD TO pay_hist_detl
EXTRACTING INVOICE-DETL      INTO ef0016; FOR LOAD TO invoice_detl
EXTRACTING INVOICE-AMT-DETL  INTO ef0017; FOR LOAD TO invoice_amt_detl
EXTRACTING MOD-DETL          INTO ef0018; FOR LOAD TO mod_detl
```

.....
Tips Program Results:

No Conversion Errors.

Files created:

SCHSQL (schema creation script)

- > should be transferred as schsql.sql

TABLOADS (data load script)

- > should be transferred as tabloads.sql

ef0001-ef0118; (data extract files)

- > should be transferred as ef#### (no extension)

USE ASCII MODE TO TRANSFER ALL FILES

END OF PROGRAM

:

As seen above, when you run TIPS, you will be asked for:

- The name of the POWERHOUSE source schema file, if any, for the flat and KSAM files you want to extract/convert. If you are doing IMAGE only, just hit return.
- The name of the TurboImage database you want to extract/convert. If you are not doing IMAGE, but only flat/KSAM files, just hit return.
- The kind of relational database you're converting to (SQL Server, Oracle, MYSQL).
- The field delimiter you want placed in the extract file, such as TAB (input as %9).

When TIPS completes, it will have created all the files you need to build your SQL database and populate it with your HP3000 data.

Output Files and Error Files

The output files are:

- schsql : Should be moved to target system as schsql.sql
- ef####: One for each extracted set / table. Automatic masters are not extracted. Should be moved to the target system as ef####.dat (ie., with the .dat extension)
- lf####: One for each extracted set/table, corresponding to the ef#### files. These are only done for Oracle target. They should be moved to the target system as lf####.ctl (i.e, with the .ctl extension)

- tabloads: For SQLServer and MySQL only (i.e., not for Oracle). Should be moved to the target system as tabloads.sql
- CONVERRS: This file defaults to \$STDLIST, but a good practice is to file equate it to a permanent disc file. (80 byte record, disc=a big number). This file contains the errors encountered in the input file, the set or file they came from, and the logical record number they occurred in.

The following kinds of errors are reported:

- Date conversion errors, if you are doing date conversion and your HP3000 contains invalid dates, they are reported here. You cannot load invalid dates to DATE type fields of SQL, because the record will not load. TIPS allows you to define a default value (like spaces or "01-01-2999") to be sent instead of an invalid date. See chapter 3 for a complete definition on how to accomplish date conversions via the CONFIG parameters.
- Numeric fields of type P and Z may contain invalid numbers. TIPS will extract any such values as 0 and report these as conversion errors. Note that it is possible to re-define the default IMAGE definition for fields, for example, extract a Z field as a character rather than a number, thus eliminating the Z field numeric conversion errors.
- If you are using the TIPS N parameter (within ITEM command, see chapter 3) to convert HP3000 ascii formatted numbers to numeric SQL types (like float, integer, etc), then any non-numeric characters (other than space, +, - and a decimal point) found in the source data will be reported as an error, and a 0 extracted instead.

Steps to Build and Populate a Database

Here is a summary of typical steps to build and populate the target database:

1. CONFIG file setup. For all numeric fields, figure out how many implied decimals they have, if any, and define them in the CONFIG file (chapter 3). If you have fields you want to load to SQL as DATE types, determine what they are, and define them, and the conversion rule you want, in the CONFIG file. Also, if you want to exclude fields or entire sets (or files), you can do so in the CONFIG file.
2. USER EXITS: If adding fields or doing any complicated data transformations, like bringing in data from other places, changing the IMAGE field type (like extracting Z items as text, not numerics) or splitting one record into 2, then user exits can be coded. See chapter 4.
3. It's a good idea to purge all the output files on the HP3000 before running TIPS. On the target system, it is also helpful to purge all these files before you move them from the HP3000, if the names exist. For example, on Unix do:

```
rm -f ef*.dat and  
rm -f lf*.ctl
```
4. Build database on target system. This involves just defining the database, without any tables, etc. For SQL Server, use the Enterprise Manager. For Oracle, you can do this with a GUI, like TOAD, or through the SQL*PLUS. The basic command is CREATE DATABASE <dbname>
5. Place the command "SET @ MAXRECS=10" (or some low number) in the CONFIG file, so that you can run TIPS quickly and check out the SCHSQL (SQL schema creation) and your UE and CONFIG stuff.
6. Run TIPS as described above. Look at conversion errors (CONVERRS) file. You might also want to PRINT one or two ef##### files just to get a feel for how your data looks, especially if you're doing date conversions or user exits.
7. Transfer all output files to target system. The script file provided with TIPS FTPSCRPT.PUB.COMP3 can be used to transfer all your ef##### and lf##### files to Unix. You should also transfer the SCHSQL file (as schsql.sql) and, for Oracle, the script found file LOADORA.PUB.COMP3, which you should send as loadora.ksh
8. On target system, using a query tool like SQL*PLUS, TOAD, or the MicroSoft Query Analyzer, execute the file schsql.sql, which you transferred. This file builds all the tables and their fields. If you have errors with this execution, it's probably due to the fact that you have SQL reserved words, like

DATE or TYPE, as your table or field names. These can be corrected in the CONFIG file with the commands:

```
SET <setname> ALIAS <a new name> Or  
ITEM @.<itemname> ALIAS <a new name>
```

If your errors are NOT due to reserved words, call or email Comp Three.

9. Load data:

For Oracle, execute the shell script loadora.ksh
For SQL Server, run the query analyzer, open and run tabloads.sql.
For more platform and SQL DBMS specific tips, see chapter 6.
After loading data, do some selects from the target db to see how the data looks.

If everything went well, remove (or comment out) the CONFIG file record

```
SET @ MAXRECS=10
```

Then purge all tables from the target DB (or just purge and recreate the empty db), then re-run steps 6 through 9. You are done! This may seem complex, but a typical database conversion, without UEs, can be done in one day. The step which is time-consuming is, obviously, running TIPS for the full extract, and loading the full extract files to the target DB.

III. CONFIGURATION (CONFIG) FILE

This file is primarily used to define numeric implied decimals, to do date conversions, and to rename sets [files] and fields which happen to be reserved words on the target db.

If you have installed TIPS, the group EXAMPLES.COMP3 contains a sample CONFIG file. CONFIG file commands can be continued on multiple lines with the & character. Comment lines begin with the asterisk (*) character. Text parameters like the alias name can be enclosed in single or double quotes. Commands are not case-sensitive. Basic commands you can place in the CONFIG file are:

SCHEMAONLY
LOADDIR
DEFAULTNUMERIC
DEFAULTREAL
DEFAULTDATE
SET
ITEM

Commands

SCHEMAONLY

Syntax: SCHEMAONLY

This command will cause TIPS to create the schema creation file (SCHSQL) but do no data extraction. This can be useful if you just want to see what the SQL layout will be, as a first step in converting your db.

LOADDIR

Syntax: LOADDIR <e.g., 'C:/temp/'>

This command is used for SQLServer and MySQL only. For these DBMS', a TABLOADS file is created which contains the commands to load your extracted data. This file references the extract file explicitly, using whatever directory you name here. Therefore, you should place in this parameter the directory you want to move the files to on your target system. Use forward slashes (not backslashes), and include the trailing forward slash, i.e., LOADDIR 'C:/temp/' not LOADDIR 'C:\temp' or 'C:/temp'.

The default for SQLServer is 'C:/comp3/'. There is no default for MySQL, so you should use this command to specify one, if you are running MySQL on a Windows machine.

DEFAULTNUMERIC

Syntax: DEFAULTNUMERIC [SQLTYPE=<e.g., FLOAT>] [NUMIMPLIED=n]

This command applies to numeric items, those with an HP3000 type of P,I,K,Z and J (note: not R or E, since these can already contain a decimal point)

By default, these items are converted for SQL with no decimal places inserted, and the SQL type of FLOAT. To override this default, you can use the DEFAULTNUMERIC command. The

DEFAULTNUMERIC definition will then be applied to all numeric except those defined with ITEM specific values for SQLTYPE or NUMIMPLIED.

For example, if your CONIG file contains:

```
DEFAULTNUMERIC SQLTYPE=MONEY NUMIMPLIED=2  
ITEM @.ORDER-QTY SQLTYPE=INTEGER NUMIMPLIED=0
```

TIPS would then convert all numeric types on the HP3000 to SQL type MONEY, with 2 decimal places inserted in the extract file fields, except for the item ORDER-QTY, which would be sent over as type integer, and no decimals applied. See ITEM command below for setting ITEM specific numeric characteristics.

DEFAULTREAL

Syntax: DEFAULTREAL [SQLTYPE=<e.g., FLOAT>] [NUMREALDECIMALS=n]

This command applies to real items, those with an HP3000 type of R or E.

By default, these items are converted for SQL with 4 significant decimal places. and the SQL type of FLOAT. To override these defaults, you can use the DEFAULTREAL command. The DEFAULTREAL definition will then be applied to all real types except those defined with ITEM specific values for SQLTYPE or NUMREALDECIMALS.

For example, if your CONIG file contains:

```
DEFAULTREAL SQLTYPE=MONEY NUMREALDECIMALS=2  
ITEM @.TAX-RATE SQLTYPE=FLOAT NUMREALDECIMALS=4
```

TIPS would then convert all real types on the HP3000 to SQL type MONEY, with 2 decimal places in the extract file fields, except for the item TAX-RATE, which would be sent over as type float, and 4 decimal places. Note that real numbers don't have "implied" decimal places, because the real number on the HP3000 already knows where the decimal place goes. The NUMREALDECIMALS just determines how many digits go after the decimal place, in the extract file. See ITEM command below for setting the ITEM specific NUMREALDECIMALS characteristic.

DEFAULTDATE

Syntax: DEFAULTDATE [sqltype=<eg, DATE, or DATETIME>]
[dateformatin=<e.g., mm/dd/yyyy, yyyyymmdd,>]
[dateformatout=<eg., mm/dd/yyyy, yyyyymmdd>]
[centurywindow=nn] *applies only to dateformatin*
[invaliddatevalue=<eg., "01/01/2099", "", >]

This command is applied to all fields you define in the CONFIG file as DATETYPE. An item can be set to DATETYPE by the ITEM command. The SQLTYPE controls the type definition for the target database. Dateformatin, dateformatout, centurywindow and invaliddate are used to drive the date conversion logic. The date conversion is done for all items which you define as DATETYPE. If you have different date formats for some items, you can override these DEFAULTDATE parameters at the item level, with the ITEM command (see below).

DATEFORMATIN describes the format of the dates on the HP3000. The allowable values are:

- Year is required, must be yyyy, yy or A0 (for MM3000 dates)

- Day is required, must be dd or ddd (day of month or day of year)
- Month is required and must be mm, if day is dd. If day is ddd, month is not allowed.
- All other characters are “passthrough”, for example, if your database contains the 8 byte value “01/01/2000”, then the dateformatin should be set to mm/dd/yyyy

DATEFORMATOUT describes the format you’d like to load to SQL. The same rules apply as per dateformatin, except A0 is not allowed (if you have an A0 type date you want to simply pass through to SQL, you should not define it as a DATEYPE, and simply let the default varchar SQL definition apply, because SQL does not recognize this datatype).

If your dateformatin has a 2 character year and your dateformatout has a 4 character year, you must supply the CENTURYWINDOW parameter. Date whose input years (2 characters) are ON OR BEFORE the CENTURYWINDOW are then converted to 20yy. Those AFTER are converted to 19yy.

In general, it’s very useful to convert to SQL datatypes, rather than simply letting dates continue to be text fields. This allows for sorting, range selection, etc. on the target system. However, invalid dates will NOT load to SQL databases. Therefore the date conversion logic must put some other value in the date. The INVALIDDATE parameter is used to replace any invalid dates. For example, if you have

```
DEFAULTDATE INVALIDDATEVALUE=""
```

Then a NULL value will be sent in the extract file for any invalid date fields. You can also use ‘01/01/2999’, or some other recognizable value which is a valid date. You should check the CONVERRS file for invalid date messages (date conversion errors) and consider correcting them on the HP3000 before migrating.

SET

Syntax: SET <setname> [maxrecs=n] [skip] [noskip] [alias <aliasname>] [pass=n]

The SET command is applied to any set or file which matches the <setname>. The setname may contain the @ character (or may be only the @ character, meaning apply to all sets and files). For purposes of clarity when converting KSAM and flat files, the command FILE is equivalent to the command SET.

MAXRECS parameter is used primarily for testing, and will limit the number of extracted records to n.

SKIP is used to entirely exclude a set or file from processing. The corresponding SQL table will not be created nor will any data extraction occur. For example, if you want to extract ONLY the dataset SALES-HEADER, you would put the following in your CONFIG file:

```
SET @ SKIP  
SET SALES-HEADER NOSKIP
```

This illustrates the wildcard feature of the setname. If multiple set parameters (in this case, SKIP and NOSKIP) apply to the same set (in this case SALES-HDR), precedence is given to the non-wildcard definition (2nd command), followed by a partial match (e.g., SET S@, which we don’t have here), followed by a “wildcard only” match (1st command). That’s why the following example works to exclude all but SALES-HDR.

ALIAS is used to cause the SQL database to have a non-default name. This parameter can only be used with an exact set name only, no wildcards. By default, TIPS will create the target table with the same name as the HP3000 name, except for the – (dash) delimiter is changed to _ (underscore), and the letters are made lowercase. For example, SALES-HEADER dataset on the HP3000 becomes sales_header table on the target SQL database.

Different SQL DBMS (Oracle, SQL Server) use different reserved words, and table and field names cannot be a reserved word. The ALIAS command gets around this by allowing you to specify any name for the target DB. For example, if you have a set called TYPE, which is reserved word on Oracle, you can use the command:

SET TYPE ALIAS TYPEX

causing the SQL tablename to be TYPEX, not TYPE.

PASS = n is used to define multiple passes through the table. A separate extract file (ef####) is created for each pass. This parameter is usually used to split records, or the entire table, as defined below in the User Exits chapter. As a brief example, suppose you wanted to split your SALES-HDR table into sales_hdr and sales_hdr_address on the target system. The sales_hdr contains all the fields except the address fields. The sales_hdr_address contains only the sales_order_no (key) and the address fields. You could do the following in the config file:

```
SET SALES-HDR PASS=1
ITEM SALES-HDR#1.ADDRESS@ SKIP
SET SALES-HDR PASS=2 ALIAS sales_hdr_address
ITEM SALES-HDR#2.@ SKIP
ITEM SALES-HDR#2.SALES-ORDER-NO NOSKIP
SALES-HDR#2.ADDRESS@ NOSKIP
```

This tells TIPS to do the first extract from SALES-HDR but exclude any field whose name begins with ADDRESS. This would go, eg., into ef0001. Then TIPS would do a second pass, creating a target tables named sales_hdr_address, containing only the key field sales_order_no, and those fields excluded in pass 1 (names starting with ADDRESS). The precedence for the ITEM name (see item command described below) is full match (no @) in name) followed by partial match (@ in name, but also other letters) , followed by “wildcard only” (item name is just @).

ITEM

Syntax: ITEM <setname>.<itemname> [#passnum] [skip] [noskip]
[alias <aliasitem>]
[type <imagetypedefinition, e.g., 2X36>]
[sqltype <sqltypedefinition>]
[keyconstraints] [nokeyconstraints]
[numimplied=n]
[numrealdecimals=n]
[datatype] [nodatatype]
[dateformatin=<e.g., mm/dd/yyyy, yyyyymmdd,>]
[dateformatout=<eg., mm/dd/yyyy, yyyyymmdd>]
[centurywindow=nn] *applies only to dateformatin*
[invaliddatevalue=<eg., 01/01/2099, “”, >]

The ITEM command is applied to any item which matches the <setname>.<itemname> description. The setname and itemname may contain the @ character (or may be only the @ character, meaning apply to all sets or items).

For example, ITEM @.ORDER-NO applies to the ORDER-NO field in any set, whereas

ITEM SALES-HEADER.ORDER-NO applied only to the ORDER-NO field in the SALES-HEADER set. As described above, the most specific item match takes precedence. Since both items and set names can have wildcards, the order of precedence, taken as a whole is:

- Exact item and set match (e.g., ITEM SALES-HDR.ORDER-NO)
- Exact item match partial set match (e.g., ITEM S@.ORDER-NO)
- Exact item match, set @ (e.g., ITEM @.ORDER-NO)
- Partial item match exact set match (e.g., ITEM SALES-HDR.@ORDER)

- Partial item match partial set match (e.g., ITEM S@.@ORDER)
- Partial item match set @ (e.g., ITEM @.@ORDER)
- Item @ exact set match (e.g., ITEM SALES-HDR.@)
- Item @ partial set match (e.g., ITEM S@.@)
- Item and set both @ (e.g., ITEM @.@)

SKIP causes the item to be excluded from the target DB. NOSKIP is primarily used to reverse the SKIP. For example, the commands:

```
ITEM SALES-HDR-ADDRESS.@ SKIP
ITEM SALES-HDR-ADDRESS.ADDRESS@ NOSKIP
ITEM SALES-HDR-ADDRESS.ORDER-NO NOSKIP
```

Will cause the target table to contain only ORDER-NO and fields whose names begins with ADDRESS.

If you are making multiple passes through a set, you can specify which pass the CONFIG record applies to by placing #<passnumber> after the set name. This is particular useful for splitting sets into two separate tables, with different field layouts (see the Advanced Topics section). If you are making multiple passes through a set but want the ITEM command to apply to all passes, just leave out the #<passnumber>.

ALIAS causes the target fieldname to assume the alias name rather than the default. This is used largely to get around items whose name is a reserved word in SQL. The ALIAS command can only be used with exact item names (the setname in the ITEM definition can still be wildcarded..., e.g., ITEM @.TYPE alias TYPEX causes the item name to be typex in the target db wherever it occurs).

TYPE causes the item's type to be whatever you want, rather than the IMAGE or POWERHOUSE definition. By default, TIPS will use the IMAGE or POWERHOUSE item type in it's conversion logic, but you can override the default type, and thus the conversion logic, with this parameter. Frequently, the IMAGE type of an item is not how it's actually used in application programs. For example, you may have an IMAGE item PHONE-NO, type Z16, but your programs may read it in as type X16, and place invalid characters (non-digits) into it. By default, TIPS will try to convert Z item types as numbers, and numeric conversion errors will result during the extract. This situation would be corrected with the following CONFIG entries:

```
ITEM @.PHONE-NO TYPE X16
```

This would cause TIPS to consider this item as if it were an X16 in the database, and it would be converted to a text (varchar) for SQL.

In addition to the IMAGE types, you can also use the type N here, which means convert the string to a numeric item. Type N is used where the data in your database is text which "looks" like a number. For example, if you have an X(10) field in your database which contains the values like 0.00, -35.12, 45.00+, etc., you can use the config command

```
ITEM <setname>.<itemname> TYPE=N10
```

to force the data to be checked, extracted, and loaded to SQL as a numeric. The default SQL type for an item you specify as an N type, is FLOAT, but you can change this with the SQLTYPE parameter, for example:

```
ITEM <setname>.<itemname> TYPE=N10 SQLTYPE=INTEGER
```

The N type causes TIPS to do the following as it extracts the field.

Strip out leading and trailing blanks.

Discard + (plus sign) character

If the – (minus) character is found, place it at the front of the extract field.

After stripping leading and trailing blanks, all character must be 0-9, . (period), - (minus sign) or +, otherwise TIPS will set the extracted field to 0 and report a conversion error for the field. If you have decimal points in your data, take care not to set the SQLTYPE to integer, but rather to something which supports decimals, by leaving it as FLOAT (ie., don't specify a specific SQLTYPE), or by explicitly setting its SQLTYPE to something like MONEY or NUMBER(15,2) <an ORACLE type>.

SQLTYPE causes the item's SQL type definition to be whatever you want, for example, SQLTYPE = varchar2 or SQLTYPE=float. As with all text parameters, the SQLTYPE value can be enclosed in single or double quotes. For example, suppose we have an item which is an X(40) on the HP3000. However, we'd like this item to be larger on the SQL database. We could say:

```
ITEM @.CUSTOMER-NAME SQLTYPE='varchar(80)'
```

causing the SQL definition to allow for 80 bytes, not 40, in the field. FYI, you can also shrink the item length, but it may then be truncated by TIPS. Also, trailing blanks are always stripped from text values.

KEYCONSTRAINTS and NOKEYCONSTRAINTS are commands which cause the IMAGE master to detail relationships to be replicated in SQL. The default is NOKEYCONSTRAINTS. If you specify KEYCONSTRAINTS for an item, then, the manual master's SQL table is build with the key as a PRIMARY KEY, which is a unique key in SQL, and the detail's SQL tables are build with a FOREIGN KEY constraints. Just like IMAGE, these characteristics mean that the manual must exist before the detail can be written, and the manual cannot be deleted if any details exist. Note that if you are specifying a MAXRECS=<n> for a manual master set, meaning you are not extracting all its records, you should use KEYCONSTRAINTS, as you will probably get a load error trying to load details, for which no master record has been loaded.

To turn on KEYCONSTRAINTS globally, you can use the command:

```
ITEM @.@ KEYCONSTRAINTS
```

NUMIMPLIED parameter only applies for numeric HP3000 items (type P, I, K,Z and J). This cause TIPS to insert a decimal point in the extracted data. For example, if you have money values in your database you may have a field in IMAGE called, for example, GROSS-INV-AMT, with a value of 123456. When this field is read into an HP3000 program, it is typically put into an implied decimal type field (like s9(09)V99, causing the program to treat it as 1,234.56. In order to get "native" SQL values correct, you must insert the decimal point before loading to SQL. By specifying NUMIMPLIED =2 for this item in the CONFIG file, TIPS will send it as 1234.56, and the SQL value will be correct.

NUMREALDECIMALS parameter is applied only to HP3000 item types R and E (real numbers). This causes the extract file to have n significant digits after the decimal place. Real numbers do not have "implied" decimal places, as the HP3000 already knows where the decimal place should go. This parameter is just for controlling how many significant digits should be sent after the decimal point, in the extract file.

DATETYPE causes the item to be treated as a date by TIPS. The date conversion logic specified in the DEFAULTDATE parameter, above, applies, and the item's value is converted per the DEFAULTDATE definition, and it's type in SQL is per the DEFAULTDATE parameters, unless overridden at the item level.

For example, a sample database had a lot of dates which were to be converted to an SQL type of DATETIME. In examining the database, all items with the word DATE in the name were date types, except for one item named UPDATE-FLAG. Therefore the following three commands in the CONFIG file solved the problem:

```
DEFAULTDATE SQLTYPE=DATETIME DATEFORMATIN=yymmdd &  
DATEFORMATOUT='yyyy-mm-dd'  
ITEM @.@DATE@ DATETYPE
```

ITEM @.UPDATE-FLAG NODATEYPE

This caused all items named @DATE@ to be converted as dates, per the date conversion specification in DEFAULTDATE, except for the item UPDATE-FLAG.

IV. USER EXITS and TIPS RUNTIME LIBRARY

Tips user exits are useful for doing more complicated transformation activities like adding new fields to tables and populating these fields dynamically, or changing values of existing fields. To write COBOL user exits, first copy the file TIPSUESK.PUB.COMP3 (for TIPS User Exit skeleton) to your local group, then add your own code, save it locally, compile and place in a local XL file. Then insert the XL in your run statement by running TIPS like:

```
RUN TIPS.PUB.COMP3;XL='<your user exit XL names>,TIPSEXL.PUB.COMP3'
```

Of course, if you want to use other local code, you can include other XLs in the list, before the last XL, which must be TIPSEXL.PUB.COMP3.

WARNING: BE SURE TO STATE THE FULL FILENAME, WITH THE GROUP AND ACCOUNT, IN YOUR USER EXIT XL NAME. OTHERWISE YOUR USER EXITS WILL NOT BE CALLED!!!

User Exits

TIPSUEInit

Called only once, when TIPS starts, allows you to fill in a 1000 byte "comarea". This can be used to open external files or databases and retain the file handle (filenumber or dbid), or set other values. The COMAREA is then passed to each TIPS UE execution for your use.

TIPSUEConvertTable

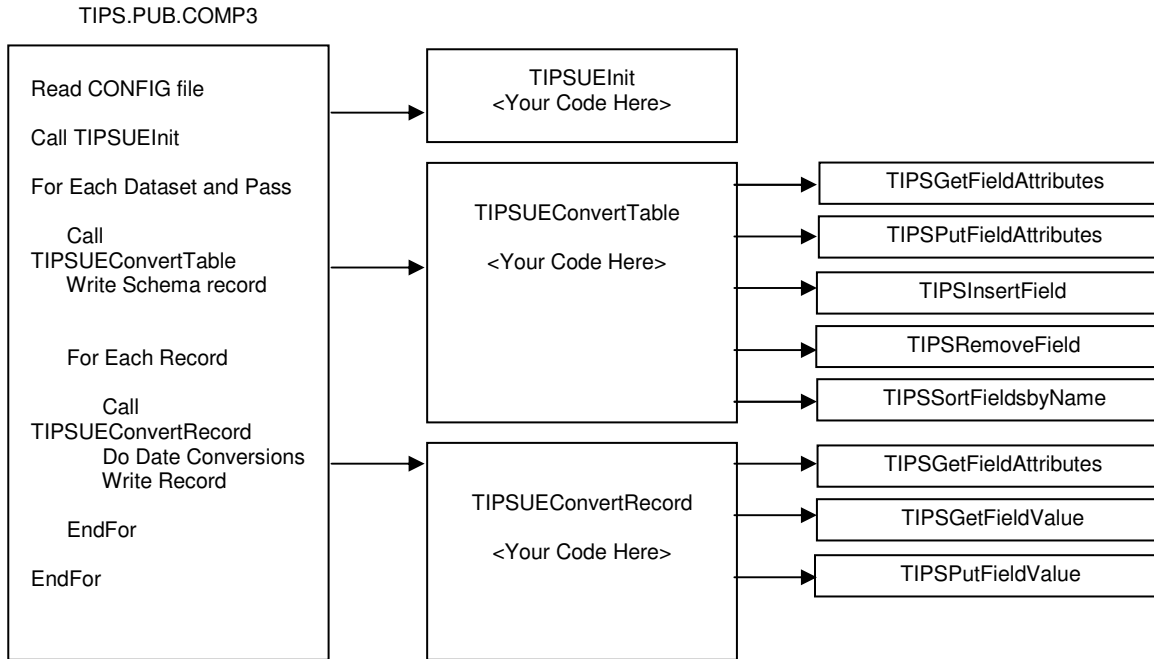
This UE is called once per dataset or file being extracted. It allows you to add fields, delete fields, or change a field's definition (e.g., change an X(40) to an X(80), so in the next UE you can put more data in the field).

TIPSUEConvertRecord

This UE is called once per record being extracted. It allows you to examine field values and change field values.

The following diagram shows the TIPS runtime structure, which UEs are called when, and what runtime procedures are available to the UEs:

TIPS Program Structure



TIPSUEInit

Parameters:

LNK-DBNAME	32 bytes	input	actual IMAGE database filename, if any. May not be modified.
UECOMAREA	1000 bytes	output	Any information you want to set and save between UE calls, like file handles

Description:

This procedure is typically used to open files. It can also be used to do anything which need only be done once per TIPS execution.

Example 1:

In this user exit, open a second database, which you will use in subsequent user-exits to populate new fields:

```

MOVE " MYDB" to MYDB-NAME.
MOVE ";" to MYDB-PASSWORD
CALL INTRINSIC "DBOPEN" USING MYDB-NAME, MYDB-PASSWORD.
MOVE MYDB-NAME (1:2) to UECOMAREA (1:2).
  
```

GO BACK.

Example 2:

A project calls for multiple HP3000 databases to be merged to a single SQL database. These databases are structurally equivalent. For example, a SALESDB exists for each local site, but we will merge them all into a single SQL database. A new variable called "SOURCE-DB" will be added in the next UE, to each record. To set the value for populating the new field, do

```
DISPLAY "What is the value you'd like for SRCDB in all records (8 bytes):"  
ACCEPT UECOMAREA (1:8)  
GO BACK.
```

TIPSUEConvertTable

Parameters:

LNK-UEAREA	1000 bytes output Any information you want to set and save between UE calls
LNK-SET-OR-FILENAME	16 bytes input only The HP3000 name of the dataset or file being extracted
LNK-ALIAS-TABLENAME	32 bytes I/O If you ALIAS'ed the SET or FILE via the CONFIG file, this is the alias Name you specified. You can set the ALIAS name here as well, by setting this parameter. This will be the tablename created in the SQL database
LNK-PASSNUM	32-bit integer input only This is the pass number, for use when you're making multiple passes through a set or file. By examining this parameter, your UE can determine which pass is being made, and do conditional logic.
LNK-SKIP-SET-FLAG	1 byte output only By setting this flag to Y or S, you can dynamically cause this set not to be extracted or defined in the schema file. This can also be accomplished with the CONFIG command SET <setname> SKIP
LNK-NUM-ITEMS	s9(09) input only The number of fields in the file or dataset you are extracting. This is the original number, before you add fields, if you choose to do so. This is useful because you can loop through all the fields to do a general task, like changing all Z types to X (see TIPSUEZ.EXAMPLES.COMP3 for an example of this).

Description

TIPSUEConvertTable executes once per extracted set or file (if you specify multiple passes, then once per pass as well). This procedure is useful for adding fields or changing field definitions. From within this UE, you can call TIPSGetFieldAttributes, TIPSPutFieldAttributes (these two are called using either a field name or a field number) TIPSInsertField, TIPSRemoveFields and TIPSSortFieldsByName.

Example 1:

You have a database which contains an item called SALES-ORDER-NO, in various datasets. In every set in which this item occurs, you want to add another, new field, called SALES-ORDER-REGION, of 40 characters, as a key value. The value of SALES-ORDER-REGION is will be determined in the next UE (TipsUEConvertRecord)by accessing another database, using SALES-ORDER-NO as the key.

In this UE, you would do the following:

```
MOVE "SALES-ORDER-NO" to FIELDNAME
CALL "TIPSGetFieldAttributes" USING  FIELDNAME
                                   FIELD-ATTRIBUTES
                                   ERR-STATUS
                                   ERR-MSG
```

If returned ERR-STATUS is 0, then SALES-ORDER-NO occurs in this dataset. Populate the FIELD-ATTRIBUTES (FA-) stuff for the new field, then insert it. For TipsInsertField, we specify where the field goes positionally, by the "before field" parameter. *FIRST and *LAST mean put the field at the beginning or end of the table. Just put the new field at the end.

```
IF ERR-STATUS = 0
    MOVE "SALES-REGION"      to FA-NAME
    MOVE "X"                 to FA-TYPE
    MOVE 40                  to FA-LEN
    MOVE "Y"                 to FA-SEARCHITEM-FLAG
    MOVE "*LAST"             to BEFORE-FIELDNAME
    CALL "TIPSInsertField" USING BEFORE-FIELDNAME
                                FIELD-ATTRIBUTES
                                ERR-STATUS
                                ERR-MSG
END-IF.
```

The population of the new field is explained below under example 1 of TipsUEConvertRecord.

Example 2:

You are doing multiple passes through a dataset, to split each record into 2 records. For example, you have a JOB-COST-DETL dataset which contains the fields: LABOR-COST, LABOR-ACCT-CODE, MATL-COST, MATL-ACCT-CODE. You would like this to be represented instead as 2 records in SQL, each with an ACCT-CODE and a COST field, (but not the other 4 fields).

In this UE, you could do the following to add the new fields COST and ACCT-CODE, with same attributes as MATL-COST and MAT-ACCT-CODE

```
IF LNK-SET-OR-FILENAME = "JOB-COST-DETL"
    MOVE "LABOR-ACCT-CODE" to  WS-FIELDNAME
    CALL "TipsGetFieldAttributes" USING  WS-FIELDNAME
                                       FIELD-ATTRIBUTES
                                       ERR-STATUS
                                       ERR-MSG

    MOVE "ACCT-CODE" to FA-NAME OF FIELD-ATTRIBUTES
    MOVE "*LAST" TO BEFORE-FIELDNAME
    CALL "TipsInsertField" USING  BEFORE-FIELDNAME
                                FIELD-ATTRIBUTES
                                ERR-STATUS
                                ERR-MSG

    MOVE "LABOR-COST" to WS-FIELDNAME
    CALL "TipsGetFieldAttributes" USING  WS-FIELDNAME
                                       FIELD-ATTRIBUTES
```

```
ERR-STATUS
ERR-MSG
MOVE "COST" to FA-NAME OF FIELD-ATTRIBUTES
CALL "TipsInsertField" USING    BEFORE-FIELDNAME
                                FIELD-ATTRIBUTES
                                ERR-STATUS
                                ERR-MSG

* Remove the 4 original fields
  MOVE "LABOR-CODE" to WS-FIELDNAME
  CALL "TIPSRemoveField" USING  WS-FIELDNAME
                                ERR-STATUS
                                ERR-MSG

  MOVE "LABOR-COST" to WS-FIELDNAME
  CALL "TIPSRemoveField" USING  WS-FIELDNAME
                                ERR-STATUS
                                ERR-MSG

  MOVE "MATL-CODE" to WS-FIELDNAME
  CALL "TIPSRemoveField" USING  WS-FIELDNAME
                                ERR-STATUS
                                ERR-MSG

  MOVE "MATL-COST" to WS-FIELDNAME
  CALL "TIPSRemoveField" USING  WS-FIELDNAME
                                ERR-STATUS
                                ERR-MSG
```

END-IF

The values of the removed fields are still available to the next UE, TipsUEConvertRecord, and the setting of their data values is explained below under example 2 of TipsUEConvertRecord.

Example 3:

You want a quick way to specify all Z fields are to be extracted as X types, because you have lots of non-numeric Z data (due to your programs treating them as X types). In this UE, you could do the following:

- LNK-NUM-ITEMS is an input parm to this UE
- WS-INDEX is declared in WORKING-STORAGE as an S9(09) COMP.
- TipsGetFieldAttributes can take either a field name (X32) or number (32-bit int) as first parm

```
PERFORM VARYING WS-INDEX FROM 1 BY 1 UNTIL WS-INDEX > LNK-NUM-ITEMS
  CALL "TIPSGetFieldAttributes" USING  WS-INDEX
                                        FIELD-ATTRIBUTES
                                        ERR-STATUS
                                        ERR-MSG

  IF FA-TYPE = "Z"
    MOVE "X" TO FA-TYPE
    CALL "TIPSPutFieldAttributes" USING  WS-INDEX
                                        FIELD-ATTRIBUTES
                                        ERR-STATUS
                                        ERR-MSG

  END-IF
END-PERFORM.
```

This example is shown in file TIPSUEZ.EXAMPLES.COMP3.

TIPSUEConvertRecord

Parameters:

LNK-UEAREA	1000 bytes output Any information you want to set and save between UE calls
LNK-SET-OR-FILENAME	16 bytes input only The HP3000 name of the dataset or file being extracted
LNK-ALIAS-TABLENAME	32 bytes I/O If you ALIAS'ed the SET or FILE via the CONFIG file, this is the alias Name you specified. You can set the ALIAS name here as well, by setting this parameter. This will be the tablename created in the SQL database
LNK-PASSNUM	32-bit integer input only This is the pass number, for use when you're making multiple passes through a set or file. By examining this parameter, your UE can determine which pass is being made, and do conditional logic.
LNK-SKIP-REC-FLAG	1 byte output only By setting this flag to Y or S, you can dynamically cause this record to not be extracted during this pass.
LNK-EXT-RECNUM	s9(09) comp input only This is the record number currently being extracted.
LNK-NUM-ITEMS	s9(09) comp. input only The number of fields in the file or dataset you are extracting. You cannot add or remove fields during this UE (must do so in the previous UE TIPSUeConvertTable)

Description

TIPSUEConvertRecord executes once per extracted record. This procedure is used for changing the value of pre-existing fields (on the HP3000) or new fields you added with TIPSUEConvertTable. From within this UE, you can call TIPSGetFieldAttributes, TIPSGetFieldValue and TIPSPutFieldValue.

Example 1:

Per Example 1 under TIPSConvertTable (above) you have placed a new field called SALES-REGION into the table. You now want to populate it by accessing another database, using the SALES-ORDER-NO as a key. In this UE, you would do the following:

```
MOVE "SALES-ORDER-NO" to FIELDNAME
CALL "TIPSGetFieldValue" USING      FIELDNAME
                                     WS-SALES-ORDER
                                     ERR-STATUS
                                     ERR-MSG
```

.....
If return-status is 0, then SALES-ORDER-NO occurs in this dataset. Access an external database you opened in TIPSUEInit, to get the value of the SALES-REGION field

```
IF ERR-STATUS = 0

    MOVE LNK-UE-DBID to MY-DBID
    CALL INTRINSIC "DBFIND" USING.....
    CALL INTRINSIC "DBGET" USING...
    MOVE "SALES-REGION"      to FIELDNAME
    CALL "TIPSPutFieldValue" USING      FIELDNAME
                                        WS-SALES-REGION
                                        ERR-STATUS
                                        ERR-MSG

END-IF.
```

Example 2:

You are doing multiple passes through a dataset, to split each record into 2 records. For example, you have a JOB-COST-DETL dataset which contains the fields: LABOR-COST, LABOR-ACCT-CODE, MATL-COST, MATL-ACCT-CODE. You would like this to be represented instead as 2 records in SQL, each with an ACCT-CODE and a COST field, (but not the other 4 fields). Per Example 2 under TIPSUEConvertTable, above, you have added the fields COST-ACCT and COST, and removed the original 4 fields. The value in the removed fields is still available to this UE. If this is PASS 1, you want to use the MATL fields as the source for the new fields. For pass 2, you will use the LABOR fields. In this UE, you could do the following:

```
IF LNK-SET-OR-FILENAME = "JOB-COST-DETL"
    IF LNK-SET-PASSNUM = 1
        MOVE "MATL-ACCT-CODE" to FIELDNAME
    ELSE
        MOVE "LABOR-ACCT-CODE" to FIELDNAME
    END-IF
    CALL "TIPSGetFieldValue" USING      FIELDNAME
                                        WS-ACCT-CODE
                                        ERR-STATUS
                                        ERR-MSG

    MOVE "ACCT-CODE" to FIELDNAME
    CALL "TIPSPutFieldValue" USING      FIELDNAME
                                        WS-ACCT-CODE
                                        ERR-STATUS
                                        ERR-MSG

    IF LNK-SET-PASSNUM = 1
        MOVE "MATL-COST" to FIELDNAME
    ELSE
        MOVE "LABOR-COST" to FIELDNAME
    END-IF
    CALL "TIPSGetFieldValue" USING      FIELDNAME
                                        WS-COST
                                        ERR-STATUS
                                        ERR-MSG

    MOVE "COST" to FIELDNAME
    CALL "TIPSPutFieldValue" USING      FIELDNAME
```

WS-COST
ERR-STATUS
ERR-MSG

END-IF

Runtime Library

The TIPS runtime library contains the procedures you call from the three UEs detailed above.:

TIPSGetFieldAttributes
TIPSPutFieldAttributes
TIPSInsertField
TIPSRemoveField
TIPSGetFieldValue
TIPSPutFieldValue
TipsSortFieldsByName
TIPSExplainError

TIPSGetFieldAttributes

Parameters:

FIELDNAME or FIELDNUMBER	x(32) or s9(09) comp input This is how you tell TIPSGetFieldAttributes which field you'd like to see the attributes for. You can pass either the field name or the field number. The field number is a just a sequential number for the field from the HP3000, or one you added via TipsInsertField. The fieldname is not case-sensitive. If you aliased the fieldname in the CONFIG file, supply the ALIAS name. Removed fields are available.
FIELD-ATTRIBUTES	record structure output only This is a record structure containing the field name, type, alias, number of implied decimals, etc. See FIELD-ATTRIBUTES record layout, below.
ERR-STATUS	s9(09) comp output only If the TIPSGetFieldAttributes fails, this is returned as non-zero. The only "normal" erroris number 1, which will have the ERR-MSG "No such field exists in this dataset/file"
ERR-MSG	x (72) output only Error message corresponding to any non-zero setting of ERR-STATUS.

Description

This procedure can be called from the UE TIPSUEConvertTable or TIPSUEConvertRecord. It returns all the FIELDATTRIBUTES parameter as shown below:

01	FIELDATTRIBUTES.	
05	FA-NAME	PIC X(16).
05	FA-FIELDNUM PIC	S9(09) COMP.
05	FA-SOURCE	PIC X(01).
	88 FA-SOURCE-IMAGE	VALUE "I".
	88 FA-SOURCE-UE	VALUE "U".
05	FA-DATETYPE-FLAG	PIC X(01).
	88 FA-DATETYPE	VALUE "Y".
05	FA-SEARCHITEM-FLAG	PIC X(01).
05	FA-EXTRACT-FLAG	PIC X.
	Y=extract data, E=extract as null (delim only)	
	S=don't extract at all; i.e. will not be in sql schema	
	88 FA-EXTRACT	VALUE "Y".
	88 FA-EXTRACT-EMPTY	VALUE "E".
	88 FA-EXTRACT-SKIP	VALUE "S".
	88 FA-SEARCHITEM	VALUE "Y".
05	FA-TYPE PIC X(01).	
	88 FA-TYPE-IS-NUMERIC	VALUES "P","I","J","K","Z".
	88 FA-TYPE-IS-REAL	VALUES "R","E".
	88 FA-TYPE-IS-TEXT	VALUES "X","U".
05	FA-FILLER	PIC X(03).
05	FA-COUNT	PIC S9(09) COMP.
05	FA-LEN	PIC S9(09) COMP.
05	FA-NBYTES	PIC S9(09) COMP.
05	FA-NUMIMP	PIC S9(09) COMP.
05	FA-NUMREALDECIMALS	PIC S9(09) COMP.
05	FA-SQLTYPE	PIC X(32).
05	FA-UE-AREA	PIC X(30).

The FIELDATTRIBUTES area is defined in the skeleton UE, TIPSUESK.EXAMPLES.COMP3, for your use.

If the field has been 'aliased' in the CONFIG file, then the FA-NAME is the alias name, otherwise it's the original HP3000 field name (from IMAGE or POWERHOUSE schema).

The FA-FIELDNUM is just a sequential number representing the field's position in the extract record.

If the FA-SOURCE is "I", the field came from the HP3000 Image, KSAM or flat file.

The FA-DATEYPE will be "Y" if you defined the field as a DATETYPE in the CONFIG file, meaning it will be undergo date conversion and be loaded to SQL as a DATE or DATETIME type.

The FA-SEARCHITEM flag will be "Y" if the field is to be converted with an index (key value). By default, key fields on the HP3000 will have this set to "Y".

FA-EXTRACT-FLAG will be set to blanks unless you've chosen to SKIP this field in the CONFIG file, or you've done a TIPSRemoveField. If the value is set to "E" the field will be built, but set to NULL (char data) or 0 during the conversion.

The FA-TYPE represents the HP3000 item type, which must be one of X, U, P, I, J, K, Z, R or E. Types X and U will be converted as text, types P, I, J, K and Z will be converted to numbers, and implied decimals applied, if specified (see CONFIG file, chapter 3). Types Z and R will be sent as floating point numbers, of the precision specified in CONFIG file (default is 4 digits after decimal).

The FA-COUNT is the number of fields in the IMAGE, KSAM or FLAT file layout, e.g., IMAGE description of 6X40 would yield a 6 here. Note that we don't load "array" fields to SQL, so an IMAGE layout like ITEM SHIP-ADDRESS 6X40 would be converted to:

```
SHIP-ADDRESS-01 varchar (40)
SHIP-ADDRESS-02 varchar(40)
...
SHIP-ADDRESS-05 varchar(40)
```

The FA-LEN is the length per the HP3000 Image definition; e.g, type X len 2 is two bytes, whereas type I len 2 is 4 bytes (32 bit integer).

FA-NBYTES is the byte length of the field, not multiplied by the COUNT. For example a 7x32 will have a 32 here, not 224.

FA-NUMIMP is the implied decimals which will be applied if the type is one of P, I, J, K, or Z (see CONFIG parm NUMIMPLIED in chapter 3).

FA-NUMREALDECIMALS is the floating point precision, i.e, number of places after the decimal for types R and E.

The FA-SQLTYPE is the sql datatype which the item will be converted to, for example, varchar(40) or MONEY or datetime.

The FA-UEAREA is 30 bytes which you can use set in TipsPutFieldAttributes, if desired, for later use within a UE.

You can retrieve, but not set this information, via the TIPSGetFieldAttributes call. For examples of its use under TipsUEConvertTable and TipsUEConvertRecord, above.

TIPSPutFieldAttributes

Parameters:

FIELDNAME or FIELDNUMBER	x(32) or s9(09) comp input This is how you tell TIPSPutFieldAttributes which field you'd like to set the attributes for. You can pass either the field name or the field number. The field number is a just a sequential number for the field from the HP3000, or one you added via TipsInsertField. The fieldname is not case-sensitive. If you aliased the fieldname in the CONFIG file, supply the ALIAS name.
FIELD-ATTRIBUTES	record structure input only This is a record structure containing the field name, type, alias, number of implied decimals,etc. Some of the fields within FIELD-ATTRIBUTES, like type, length, etc. can be changed (see below).
ERR-STATUS	s9(09) comp output only If the TIPSPutFieldAttributes fails, this is returned as non-zero. The only "normal" error is number 1, which will have the ERR-MSG "No such field exists in this dataset/file".
ERR-MSG	X (72) output only

Error message corresponding to any non-zero setting of ERR-STATUS.

Description

This procedure can only be called from the UE TIPSUEConvertTable user exit. You can use it to change settings of existing fields within the FIELD-ATTRIBUTES. The fields you may change are:

- FA-NAME (will cause the sql schema name to be whatever you supply)
- FA-DATEYPE (may be set to Y or N)
- FA-EXTRACT-FLAG (may be set to Y, S, or E)
- FA-TYPE (may be set to one of X, U, P, I, J, K, Z, R or E)
- FA-SEARCHITEM-FLAG may be set to “Y” or “N”, resulting in an index being created (or not) in the SQL database. The default is “Y” for HP3000 keys.
- FA-LEN may be set to any native HP3000 length for that type. If a type is made shorter, data truncation will occur for text items, and for numeric fields, the field’s value will be unpredicable and must be via the TipsPutFieldValue call. If the length is made longer, the field on the SQL database will be larger. If set to longer for a numeric or Z type, the field’s value will be unpredicable and must be set via the TipsPutFieldValue call.
- FA-COUNT may be set to any value > 0. If increased, new fields will be created in SQL (with the names appended with -01, -02, etc) . but these fields will be set to NULL (char data) or 0 unless set with the TIPSPutFieldValue call.
- FA-SQLTYPE may be set to any string, e.g., “varchar (40)”, and the SQL schema will be created with that item type. Invalid SQL item types are not filtered out, and will result in SQL Schema creation errors.
- FA-NUMIMP may be set to any non-negative value. The default value is as per the CONFIG file specification. This is only used for items of type P, I, J, K and Z. This is the value that causes the implied decimals to be supplied in the extract file.
- FA-NUMREALDECIMALS may be set to any non-negative value. The default value is as per the CONFIG file specification. This is only used for items of type R and E. This is the value that causes the floating point number of decimals of precision to be supplied in the extract file.
- FA-UEAREA may be set to anything, and can be used to communicate data, e.g., between the UEs TIPSUEConvertTable and TIPSUEConvertRecord.

Setting any other FA fields has no effect, and is ignored by TipsPutFieldValue. For examples of TipsPutFieldValue use, see example 3 under TipsUEConvertTable example above or see the files TIPSUEEG.EXAMPLES.COMP3 and TIPSUEZ.EXAMPLES.COMP3.

In practical use, the only thing you can do with this UE that you can’t do with CONFIG file parameters, is increasing or decreasing the length of an existing field. Therefore this procedure is rarely used. Example 3 in TipsUEConvertTable, above, could also have been accomplished via making a CONFIG file record of

```
ITEM @.<itemname> TYPE=X<len>
```

for every Z item in your database/file(s). TIPSUEEG.EXAMPLES and TIPSUEZ.EXAMPLES show sample code using this procedure.

TIPSInsertField

Parameters:

BEFORE-FIELD x(32) input only

	This is how you tell TIPSInsertField where you want the new field placed. The new field will be placed before the field specified here. The values “*FIRST” and “*LAST” are also accepted, and, in practice, “*LAST” is usually used here.
FIELD-ATTRIBUTES	record structure input only This is a record structure containing the field name, type, alias, number of implied decimals, etc. See explanation under TipsGetFieldAttributes, above.
ERR-STATUS	s9(09) comp output only If the TIPSInsertField fails, this is returned as non-zero.
ERR-MSG	X (72) output only Error message corresponding to any non-zero setting of ERR-STATUS.

Description

TipsInsertField is used to add a new field to the table as loaded to SQL. Before calling TipsInsertField, you must set, at a minimum, the FA-NAME, FA-COUNT, FA-TYPE, FA-LEN, FA-SEARCHITEM-FLAG, FA-DATATYPE-FLAG, and FA-EXTRACT-FLAG. If you are inserting a P type field, you must also set FA-NUMIMP (number of implied decimals). If you are inserting an R or E item, you must set FA-NUMREALDECIMALS (number of digits of precision after decimal point).

You may optionally set FA-UEAREA and FA-SQLTYPE. SQLTYPE will be determined for you if you don't supply it.

Setting FA-NBYTES, FA-FIELDNUM and FA-SOURCE has no effect, since TIPSInsertField does this for you.

Type, Len and Count should conform to IMAGE acceptable values, although COUNT and LEN can exceed IMAGE maximums, and are limited by the maximum record length considering all pre-existing, inserted and removed fields (which is 100,000 bytes). This procedure will also return an error if you attempt to place more than 1000 fields in a table, including all pre-existing, inserted and removed fields. Examples of this procedure are shown above under TipsUEConvertTable.

TIPSRemoveField

Parameters:

FIELDNAME	x(32) or s9(09) comp input only This is how you tell TIPSRemoveField which field to remove. The field name or its sequential number may be supplied. The fieldname is not case-sensitive. If you aliased the fieldname in the CONFIG file, supply the ALIAS name.
ERR-STATUS	s9(09) comp output only If the TIPSRemoveField fails, this is returned as non-zero.
ERR-MSG	X (72) output only Error message corresponding to any non-zero setting of ERR-STATUS.

Description

TipsRemoveField is used to remove fields from the target SQL table. The removed fields will not be in the schema for the SQL table and will not be extracted. However, you can examine the value of the removed field, during the TipsUEConvertRecord call. See examples under TIPSUEConvertRecord, above.

TIPSGetFieldValue

Parameters:

FIELDNAME	x(32) or s9(09) comp input only This is how you tell TIPSGetFieldValue which field you want the value of. The field name or its sequential number may be supplied. The fieldname is not case-sensitive. If you aliased the fieldname in the CONFIG file, supply the ALIAS name.
FIELDVALUE	output only. Supply a target buffer large enough to contain the value of the requested field, just as you would a DBGET target. If the field contains a count, then the entire array is returned. For example, if the requested field is 6x72, 452 bytes will be returned.
ERR-STATUS	s9(09) comp output only If the TIPSGetFieldValue fails, this is returned as non-zero.
ERR-MSG	X (72) output only

Description

This procedure is callable only from TipsUEConvertRecord, and is used to examine the value of one field for the record currently being extracted. Fields which you removed are also available for examination. See examples under TipsUEConvertRecord, above.

TIPSPutFieldValue

Parameters:

FIELDNAME	x(32) or s9(09) comp input only This is how you tell TIPSPutFieldValue which field's value to set. The field name or its sequential number may be supplied. The fieldname is not case-sensitive. If you aliased the fieldname in the CONFIG file, supply the ALIAS name.
FIELDVALUE	output only. Supply a target buffer which holds the entire contents of the field, just like a DBPUT. If the field contains a count, then the entire array must be supplied. For example, if the field being put is 6x72, supply 452 bytes of data.
ERR-STATUS	s9(09) comp output only If the TIPSPutFieldValue fails, this is returned as non-zero.

ERR-MSG X (72) output only

Description

This procedure is callable only from TipsUEConvertRecord, and is used to change the contents of one field for the record currently being extracted. See examples under TipsUEConvertRecord, above.

TIPSSortFieldsByName

Parameters:

ERR-STATUS s9(09) comp output only
If the TIPSSortFieldsByName fails, this is returned as non-zero.

ERR-MSG X (72) output only

Description

This procedure is callable only from TipsUEConvertTable. It causes the resultant SQL table to be sorted by fieldname. If you want the fields on the SQL database to be in exactly the same order as you are used to on the HP3000, you would not use this. However, it can be beneficial to take this opportunity to arrange the fields in sorted name order in SQL, as it makes things easier to find.

TIPSExplainError

Parameters

None

Description

TipsExplainError shows the last error which occurred in any TIPS runtime procedure library call. You can also just DISPLAY the ERR-STATUS and ERR-MSG after each non-succesful call yourself, if you prefer.

V. HELPFUL HINTS

Sample Files

The EXAMPLES group of the COMP3 account contains a sample configuration file, name CONFIG, and some sample user exit programs, whose names begin with TIPSUE.

Be sure to start with one of the sample user exit programs before coding your own UEs, as they contain the record structures you will need to invoke the runtime library procedures.

The file FTPSCRPT.PUB.COMP3 is a generic ftp script which recognizes and sends all TIPS created files to a parameter supplied IP address target. You can use this script to transfer your files.

Conversion Errors

It is very common to have conversion errors during your extract, so be sure to start with a small number of records to see the issues (CONFIG file parm SET @ MAXRECS=). Many or most large production databases have invalid dates, line-feeds embedded in text data, invalid characters in IMAGE Z type fields, and invalid P type numeric data. All of these cause TIPS to convert the source data before extracting and sending, as follows:

- LF's in text data are replaced with spaces.
- Invalid numbers in P and Z fields cause the field to be sent as 0.
- Invalid dates cannot load to SQL date fields, so they are sent as NULLS (default), or as specified in your CONFIG file INVALIDDATEVALUE parameter.

You have a few choices in dealing with these conversion errors:

- Do nothing. The data, like a date of "99990000" makes no sense anyway, so let it be sent as NULL. Or a numeric P field with junk cannot be fixed, so let it be sent as 0.
- Fix the data on your HP3000 before re-running TIPS. For example, change a date of 00/00/2000 to 01/01/2000.
- Change the type of the field via the CONFIG file. For example, many people define Z fields in IMAGE but store non-digits in them. Their application programs read these fields into X field types, so they never notice until it's time to convert these fields to numbers and load to SQL. Using the TIPS CONFIG command ITEM <setname>.<itemname> TYPE= X<numbytes>, you can cause the type to be item to be extracted and sent as a text type, which matches how you were using it on the HP3000. For dates, if you don't set the DATETYPE parm in TIPS for the field, the dates will be sent ala their HP3000 definition, e.g., X8 go over as varchar(08), and invalid dates can thus be loaded to SQL.

Rerunning TIPS

It almost never makes sense to delete records from the SQL database via the delete command. The usual method is to either

- drop all tables
- purge then re-create the empty database

Then re-execute the schsql schema creation file built by TIPS and reload the data.

SQL SERVER TIPS

For SQL Server, be sure to use transfer type ASCII if transferring to Windows via the Reflections File Transfer functionality. You can use the TIPS supplied file FTPSCRPT.PUB.COMP3 to send the data via ftp.

You should set up a directory called c:\comp3 on the Windows system, and send the files there. This is where the load script TABLOADS expects the files to be.

The SQL Server fileset is:

```
SCHSQL (should be sent to C:\COMP3\schsql.sql)
TABLOADS (should be sent to C:\COMP3\tabloads.sql)
Ef##### (should be sent to C:\COMP3\ef#####, no file extension)
```

The Enterprise Manager should be run to create the “empty” database, with no tables. Then the Query Analyzer should be run to:

1. Open and execute c:\comp3\schsql.sql
2. Open and execute c:\comp3\tabloads.sql

SQLServer Dates

To migrate to SQL Server datatypes, use the DATEFORMATOUT parm of yyyy-mm-dd or yyyymmdd, and the SQLTYPE of datetime For example:

```
DEFAULTDATE dateformatin=<your HP3k date fmt> &
dateformatout=yyyy-mm-dd SQLTYPE=datetime
```

or

```
ITEM @.INV-DATE DATETYPE &
dateformatin=<your HP3k date format> &
dateformatout=yyymmdd SQLTYPE=datetime
```

UNIX / ORACLE TIPS

On Unix, you should create a new directory to hold all the TIPS files (we'll refer to it as the TIPS directory). Copy the file LOADORA.PUB.COMP3 to this directory as loadora.ksh, and make it executable with a command like `chmod 666 loadora.ksh`. The files created by TIPS, which must be moved to this directory after TIPS execution, are:

```
SCHSQL (should be sent as schsql.sql)
ef##### (should be sent as lowercase names, with the .dat extension)
lf##### (should be sent as lowercase names, with the .ctl extension)
```

These files can easily be moved with the FTPSCRPT.PUB.COMP3 command file.

Then the empty database, with no tables, should be created by the DBA.

If re-doing the extract, you now drop all the tables from the database, using a tool like TOAD, or SQL*PLUS, and also purge all the files from the TIPS directory, with the `rm` command

```
rm -f *.sql
rm -f lf*.ctl
rm -f ef*.dat
```

Next you re-run TIPS and transfer all files to the TIPS directory, using the supplied script file FTPSCRPT.PUB.COMP3

Next log onto the Unix, chdir to the TIPS directory, and execute the command:
`./loadora.ksh <dbname> <username> <password> 2>&1 > loadora.log`

supplying the databasename, username, and user password as setup by the DBA.

This script will find all ef####.dat files and load them, using the lf####.ctl files, to the appropriate tables.

See the comments at the top of LOADORA.PUB.COMP3 for more details. The command
`grep -n due loadora.log`

shows a summary count of any load errors which occurred, by lf#### file number.

To see specific load error for a specific table, use:
`cat lf####.log`

using the specific lf file number, like lf0001, as shown in the grep command.

Finally, to see the actual records which did not load to SQL (if any), use command
`cat ef####.bad`

again supplying the actual ef file number, like ef0001.

Oracle Dates

To migrate to ORACLE date types, use the SQLTYPE of DATE. If your dateformatout is other than YYYY-MM-DD or YYYYMMDD, also specify this in your SQLTYPE, in quotation following the word DATE, as follows:

```
DEFAULTDATE dateformatin=<your HP3K date format> &  
DATEFORMATOUT=mm/dd/yyyy &  
SQLTYPE="DATE 'mm/dd/yyyy'"
```

as with all TIPS CONFIG parameters which are text types, you can use single or double quotes around the parameter. In this case, since ORACLE will require the date format to be quoted, use double quote around the TIPS SQLTYPE parameter, and single quotes within the parameter to delineate the Oracle date format

As stated above, if your output format is YYYYMMDD or YYYY-MM-DD, you need only specify DATE in the SQLTYPE, as follows:

```
DEFAULTDATE dateformatin=<your HP3K date format> &  
DATEFORMATOUT=yyyy-mm-dd &  
SQLTYPE="DATE"
```

VI. Sample CONFIG FILE

*Lines beginning with an asterick are comments, not processed by TIPS

*SCHEMAONLY

*SET @ MAXRECS=1

DEFAULTDATE DATEFORMATIN=YYMMDD DATEFORMATOUT=YYYY-MM-DD &
INVALIDDATEVALUE=" CENTURYWINDOW=60 SQLTYPE=DATE

ITEM @.DATE@ DATETYPE

ITEM @.DATE-SHIPPED NODATETYPE

* ITEM STUFF

DEFAULTNUMERIC NUMIMPLIED 2 SQLTYPE=NUMBER(15,2)

*DEFAULTNUMERIC NUMIMPLIED 2 SQLTYPE=MONEY

* GENERAL (ALL DATASET) NUMBERS <> DEFAULTNUMERIC

ITEM @.AR-PRCNT NUMIMPLIED 12 SQLTYPE=FLOAT

ITEM @.LOE-FEE-RATE NUMIMPLIED 3 SQLTYPE=FLOAT

ITEM @.LOE-PRCNT NUMIMPLIED 1 SQLTYPE=FLOAT

ITEM @.PHYS-COMP-RATE NUMIMPLIED 8 SQLTYPE=FLOAT

ITEM @.FIXED-FEE-RATE NUMIMPLIED 8 SQLTYPE=FLOAT

ITEM @.SUB-CONT-HRS NUMIMPLIED 1 SQLTYPE=FLOAT

ITEM @.ALLOC-PRCNT NUMIMPLIED 4 SQLTYPE=FLOAT

ITEM @.CPR-RATE NUMIMPLIED 12 SQLTYPE=FLOAT

ITEM @.WITHHOLD-PRCNT NUMIMPLIED 12 SQLTYPE=FLOAT

ITEM @.VAT-RATE NUMIMPLIED 4 SQLTYPE=FLOAT

ITEM @.HOURS NUMIMPLIED 1 SQLTYPE=FLOAT

ITEM @.PSUB-CONT-HRS NUMIMPLIED 1 SQLTYPE=FLOAT

ITEM @.GA-RATE NUMIMPLIED 3 SQLTYPE=FLOAT

ITEM @.ENTITY-PRCNT NUMIMPLIED 12 SQLTYPE=FLOAT

ITEM @.LINE-15-RATE NUMIMPLIED 12 SQLTYPE=FLOAT

ITEM @.INTEREST-RATE NUMIMPLIED 3 SQLTYPE=FLOAT

ITEM @.FEE-PAYABLE-RATE NUMIMPLIED 6 SQLTYPE=FLOAT

ITEM @.UNIT-LIQ-AMT NUMIMPLIED 5 SQLTYPE=FLOAT

ITEM @.LINE5-PRCNT NUMIMPLIED 12 SQLTYPE=FLOAT

ITEM @.LOSS-RATIO NUMIMPLIED 10 SQLTYPE=FLOAT

ITEM @.ENTITY-GA-RATE NUMIMPLIED 3 SQLTYPE=FLOAT

ITEM @.QTY NUMIMPLIED 0 SQLTYPE=INTEGER

ITEM @.BILL-PERIOD-CP NUMIMPLIED 0 SQLTYPE=INTEGER

ITEM @.BILL-PERIOD-FP NUMIMPLIED 0 SQLTYPE=INTEGER

ITEM @.BILL-PERIOD-PP NUMIMPLIED 0 SQLTYPE=INTEGER

ITEM @.CONT-SEQ-NO NUMIMPLIED 0 SQLTYPE=INTEGER

* SET SPECIFIC NUMBERS <> NUMIMPLIED 2

* UNIT-PRICE

*PBP-DETL.UNIT-PRICE NUMIMPLIED 2 SQLTYPE=NUMBER(15,2)

ITEM PBP-DETL.UNIT-PRICE NUMIMPLIED 2 SQLTYPE=NUMBER(15,2)

ITEM ITEM-HIST-DETL.UNIT-PRICE NUMIMPLIED 5 SQLTYPE=FLOAT

ITEM ITEM-DETL.UNIT-PRICE NUMIMPLIED 5 SQLTYPE=FLOAT

* CMF-RATE, POH-RATE AND CMF-RATE

ITEM RATE-DETL.CITE-CMF-RATE	NUMIMPLIED 5	SQLTYPE=FLOAT
ITEM RATE-DETL.POH-RATE	NUMIMPLIED 3	SQLTYPE=FLOAT
ITEM RATE-DETL.CMF-RATE	NUMIMPLIED 5	SQLTYPE=FLOAT
ITEM WIP-YTD-TOT-DETL.CITE-CMF-RATE	NUMIMPLIED 4	SQLTYPE=FLOAT
ITEM WIP-YTD-TOT-DETL.POH-RATE	NUMIMPLIED 4	SQLTYPE=FLOAT
ITEM WIP-YTD-TOT-DETL.CMF-RATE	NUMIMPLIED 3	SQLTYPE=FLOAT