 TRIM3 Simulation Technical Reference

CPS Conversion

**I.** [**Introduction**](#Intro)

**II. What You Need Before You Begin**

1. [**CPS Public Release Data File and Technical Documentation**](#TwoOne)
2. [**MySQL Database Objects**](#TwoTwo)
3. [**SAS Pre-processing codes**](#TwoThree)
4. [**Consumer Price Index**](#TwoFour)
5. [**Monthly Employment and Unemployment Statistics**](#TwoFive)
6. [**Pass-through Child Support Values for Each State**](#TwoSix)
7. [**TRIM3 TrimEXE C++ workspace**](#TwoSeven)

**III. Prepare the TRIM3 Database for Conversion**

1. [**Background Information on the TRIM3 Database**](#ThreeOne)
2. [**Create a New Input Schema for the Current Year**](#ThreeTwo)
3. [**Prepare the CTD.Map Table for the New Schema**](#ThreeThree)
4. [**Review the CPS Technical Documentation and Update the CTD.Map and TRIM3 Code**](#ThreeFour)
5. [**Update the CTD.Terms Table**](#ThreeFive)
6. [**Update the TRIM3 TrimDLL and Convert Projects for the Current Data Year**](#ThreeSix)
7. [**Perform SAS Pre-processing of the Input Data**](#ThreeSeven)

**IV. Run the Conversion**

1. [**Run Convert via the TRIM3 Interface**](#FourOne)
2. [**Conversion Data-flow Diagram**](#FourTwo)

**V. Verify Conversion and Create a New TRIM3 Input File**

##### [**Perform an Initial Check of the Data**](#FiveOne)

1. [**Review the Convert Log**](#FiveTwo)
2. [**Perform Required Manual Database Edits**](#FiveThree)
3. [**Make the New Schema Accessible to Registered TRIM3 Users**](#FiveFour)
4. [**Refresh the Database on Cybele**](#FiveFive)
5. [**Generate Term Statistics**](#FiveSeven)
6. [If Required **Correct the Microdata.Variable Table for the Current Year**](#FiveSeven)
7. [**Generate and Test a New Make File**](#FiveEight)

**VI.** [**Post Conversion Tasks**](#Six)

**VII.** [**Conversion Process Diagram**](#Diagram)

**VIII.** [**Appendix A: Procedure for Adding a New CPS Variable
To Input Schema Tables Only**](#AppendixA)

**I. Introduction**

This document describes the process of turning a raw CPS file into a TRIM3 input database. It may be useful to a number of different people but is particularly important for a programmer who is performing a conversion and input data verification.

###### II. What You Need Before You Begin

## II.1. CPS Public Release Data File and Technical Documentation

You will need the CPS March survey Annual Social and Economic Supplement (ASEC) for the new survey year. It can be downloaded from the Census ftp site at <http://www.bls.census.gov/ferretftp.htm>. To download, click on the CPS March supplement link and select year.

You may also download the data dictionary, though this is optional, as it is not used for conversion. Rather, refer to the more comprehensive, and likely more reliable, Technical Documentation, available from <http://www.census.gov/apsd/techdoc/cps/cps-main.html>. Alternatively, this site may be reached by going to the Census Bureau’s home page at <http://www.census.gov/>, clicking on the “Subjects A to Z” link at the top of the page, then “C,” then “Current Population Survey (CPS),” then, on the left side of the page, the “CPS Technical Documentation” link.

**II.2. MySQL Database Objects**

# The CTD.InputSchemas Table

The InputSchemas table contains the name of every input schema recognized by TRIM3 on a given server.

# The CTD.Map Table

The CTD.Map table is a bridge between the variables on the raw CPS and the variables in a TRIM3 input schema. CTD.Map defines a set of variables for each input schema, with one record for each variable.

# The CTD.Terms Table

The CTD.Terms table is the main data dictionary table for TRIM3. Any new CPS variables which you intend to add to the TRIM3 input data must be defined in CTD.Terms before TRIM3 will be able to reference them.

# The Microdata.Variables Table

The Microdata.Variables table contains one record for each variable in a schema and is the controlling table used by the Microdata Examiner and the TrimDLL project’s Make process. This table is automatically re-populated when a database is refreshed. However, it is possible that manual modifications could be required prior to creating a Make file for a new or existing schema in case of an initial Make file creation error.

## II.3. SAS Pre-processing codes

The TRIM3 Convert project is unable to read a March ASEC public release file as distributed, so some SAS pre-processing is required. **Be sure to obtain the latest versions of SAS codes used for conversion from the most recent subfolder in the Cybele/source/Convert, SAS Source and Data folder. Once conversion is successfully completed, save the updated files to a new subfolder for the current data year**.

If you are performing an initial conversion of March ASEC data without incorporating immigrant imputations and splitting household, then only one SAS code (RecodeTeenHeads.sas) must be run prior to preparing the database for the new data and running the TRIM3 convert process. This code is also the first step of a default conversion which incorporates the immigrant imputations and splits some immigrant and high-income households.

RecordTeenHeads.sas restructures CPS households that are headed by a young, single householder who lives with at least one parent. In households that are identified for restructuring, the CPS family structure and household and family relationship codes are recoded so that a parent is designated as the householder. This SAS code produces the input dataset that must be used for the remaining conversion steps. Be sure to use the last version of the code used for conversion in case it has been recently updated. The latest versions of SAS conversion codes are copied to the source folder of the ASPE-TRIM server at the completion of every conversion.

If you need to clone high-income households for the SOI match, then you will need the <IdentifyHighIncomeHHs.sas> and [Clone[Year].sas](Clone2005.sas) SAS codes.

If you are performing an immigrant conversion, you will need the <CheckImmigrantData.sas> and [ImmigrantMerge[Year].sas](ImmigrantMerge2005.sas) codes and a data file containing the Urban Institute immigrant imputations which will typically be provided by a researcher in IBP or LHP. The imputed immigrant data file should contain the following fields:

CITSTAT– Imputed immigrant status

H\_SEQ – ASEC household sequence number

ITERATN – Final iteration number (for cloned households; zero indicates a household was not cloned)

LINENO – ASEC person line number

PPPOS – ASEC person sequence number

UILEG1 – Recode of CITSTAT

UI\_WGT – New person weight that replaces the original ASEC weight in cloned households

**II.4.** **Consumer Price Index**

Obtain the data year’s CPI from [www.bls.gov](http://www.bls.gov). Click on “Inflation and Prices,” then “Consumer Price Index.” Scroll down the page to “All Urban Consumers (Current Series)” and click on the “Tables” link, then the Consumer Price Index History Table, “All Items Indexes and Annual Percent Changes From 1913 to Present.”[[1]](#footnote-1) Find the CPI in the Annual Average column for the current year and use that value to compute the latest value for the CPI array in the TrimEXE commun.cpp file. (See calculation instructions in the commun.cpp code just prior to the CPI array that needs to be updated.) Also remember to increment array size in commun.h. Compute values for two or three former years and compare with the values in the CPI array for those years to ascertain you have obtained the correct data and have used the right procedure to calculate the new CPI array value.

**II.5.** **Monthly Employment and Unemployment Statistics**

Obtain the data year’s monthly employment and unemployment statistics from [www.bls.gov](http://www.bls.gov). We use historical data from the ‘A’ Tables, employment and unemployment status for the total civilian population (from tables showing statistics by sex and age), not seasonally adjusted.

As the BLS site becomes increasingly difficult to navigate, it may be helpful to include here the link where this information was last found. For the 2008 conversion, these data were located at <http://www.bls.gov/cps/cpsatabs.htm>. Select Table A-1 and check two boxes (employed and unemployed) for total civilians, not seasonally adjusted. Then click “Retrieve Data” at the bottom of the screen.

Retrieve data for both the data year being converted and the prior year so you can verify the data with existing values in the BlsEstimatedPersonsWorking and BlsEstimatedPersonsUnemploy arrays of Convert/Allocate.cpp.

## II.6. Pass-through Child Support Values for Each State

Obtain these values from the Welfare Rules Database at <http://anfdata.urban.org/wrd/Query/query.cfm> or from the IBP analyst who is responsible for the TRIM3 ChildSupport module.

## II.7. TRIM3 TrimEXE C++ workspace

Obtain the latest TrimEXE workspace from the CVS repository or from Cybele Source.

### III. Prepare the TRIM3 Database for Conversion

##### **III.1 Background Information on the TRIM3 Database**

The CTD schema contains tables that control much of the processing performed by the TRIM3 Engine, often referred to as the “frame”, by Convert, and by simulation code. CTD is an acronym for “Central Trim Database.”

Input schemas are parallel to the CTD. Each input schema contains seven tables created by the TRIM3 Engine’s Convert process: Household, Family, Person, Adult, AdultCurrent, AdultPrevious, and AdultMonthly. Input schemas also contain TermStatistics tables that contain statistics on each of a schema’s variables. The TRIM3 Engine recognizes only the following input schema prefixes: “CPS,” “CPSI,” “NSAF,” and “TEST” (case insensitive). The “I” following “CPS” refers to a CPS schema with UI immigrant imputations and split households. The last four characters of every schema name are digits that refer to survey year. Note that survey year and data year differ by 1. A March 2006 CPS survey collects information on data year 2005.

**III.2. Create a New Input Schema for the Current Year**

Using a MySQL tool such as Control Center or Browser, create a new schema with the command “Create database <input schema name>.” Then open the CTD schema and add a record to the InputSchemas table for the new schema following established conventions. The InputSchemas table contains the following fields.

|  |  |
| --- | --- |
| **Field** | **Description** |
| *Schema* | The name of an input schema, e.g., “CPSI2006” |
| *Year* | The data year that will be used to identify the schema in TRIM3 database tables. Warning: This year must be unique. For the CPSI2006 input schema, year is “Alien2005.” |
| *Description* | A brief description that identifies the schema to users that will also be displayed by the TRIM3 interface. |
| *SimulationYear* | A 4-digit year that corresponds to a schema’s data year. SimulationYear for the CPSI2006 schema is “2005.” This field does not have to be unique. There may be multiple input schemas for a single data year. |
| *Version* | A brief version identifier that will be displayed by the TRIM3 interface by which users will select a schema. Since immigrant schemas are the default data source in TRIM, the CPSI2006 input schema version is “\*\*\*Default\*\*\*.” |
| *Sequence* | A version sequence number which determines the order in which schema versions are shown by the TRIM3 interface. |

The first three or four characters of schema name must be either “CPS,” “CPSI,” “NSAF,” or “TEST” (case insensitive). The last four characters must be digits that refer to survey year.

**III.3. Prepare the CTD.Map Table for the New Schema**

The CTD.Map table is a bridge between the variables on a raw CPS data file and the variables in a TRIM3 input schema. It informs TRIM’s Convert process of the fields to be included in a new schema and the location and length of each field on the ASCII input file read by Convert. The CTD.Map table contains the following fields.

|  |  |
| --- | --- |
| **Field** | **Description** |
| *MapID* | The name of the input schema to which the map record applies. |
| *RecordType* | Controls reading of hierarchical input data during conversion. 1=Household; 2=Family; 3=Person; 4=A field not on an input file that is generated by Convert. |
| *TRIM3Name* | TRIM3 variable name. |
| *Col* | Starting position on the ASCII input file. |
| *Lenght* | Number of characters a variable takes up on an input file. “Lenght” is not a typo; the column name itself is misspelled. |
| *RangeLow* | Lowest valid value for this variable. |
| *RangeHigh* | Highest valid value for this variable. |
| *CPSName* | CPS variable name. |
| *LongName* | May contain a descriptive name or variable definition. Not needed for conversion. |
| *Universe* | CPS universe. Not needed for conversion. |
| *Enumerate* | Name of enumerator that contains field values. Not needed for conversion. |
| *Status* | Not needed for conversion. |
| *TableName* | Name of the table in the TRIM3 input schema where a variable is to reside. Not needed for conversion. |

Enter one map record for each variable on a Schema’s input file (RecordType 1-3) or that—in years prior to the 2012 ASEC—are to be generated by Convert (RecordType 4). Convert will ignore any map records for which no TRIM3 name is provided. To facilitate the process of adding new map records, create a temporary table with the same structure as CTD.Map. Then copy the CTD.Map records used for a similar prior conversion to the temporary table, rename the mapid and make other required edits to the records in the temporary table using MySQL commands and a MySQL tool such as Control Center or Browser. Once all required changes, deletions, and additions have been made in the temporary table for the new schema, copy the edited records to the CTD.Map table.

**ALERT**: Beginning with the conversion of 2010 ASEC data, we added a new conversion code that alters the family structure of some households with teen heads. This code (RecodeTeenHeads.sas) is run as the first conversion step for both an initial conversion and a default conversion. Ascertain that the positions of all the variables that are written to the restructured file agree with the locations specified in the map table and, for a default conversion, also agree with the locations read by other SAS codes that follow.

Type 4 Records (Discontinued)

Type 4 records are no longer used. Beginning with the 2012 ASEC data, the conversion code was modified to eliminate the need for these records. The information in the paragraph below pertains to earlier conversions when they were required.

TRIM3 names in the ctd.map table for all Type 4 records begin with the letter “C,” which refers to specific C++ conversion classes. Each Type 4 record invokes the execution of the class of functions in the conversion code corresponding to the TRIM3 name. Ordering of the Type 4 records in the CTD.Map table is, thus, significant. A Type 4 record that causes convert to execute coding to compute DetailedOtherIncome, for example (the Type 4 record with TRIM3 name “CDetailedOtherIncome”), must be executed after coding that computes the components of DetailedOtherIncome—i.e., LastOwnWorkersCompIns, LastOwnDisabilityIns, LastTotalEducationalAssistance, LastBlackLungMinerBen, LastStateTempDisability and LastAnyOtherIncome. Since it is difficult to ascertain record ordering in the map table, you can ensure that a value such as DetailedOtherIncome is calculated after its components have been calculated by adding it to the map table only after its components have been added.

Map Records that are Unique to Default Conversions

A conversion that clones high-income households requires a CTD.Map record for the household-level TRIM3 field HighIncomeClone that is not required for conversions without the cloning. And an immigrant conversion requires five additional CTD.Map records for household-level AlienHouseholdSplit, and person-level variables CorrectedCitizenship, DetailedCorrectedCitizenship, EditedCountryOfBirth and EditedYearInUS. Additionally, if the input file being used for conversion has been edited to restructure inverted households[[2]](#footnote-2), another map record is needed for the person variable ASECFamilyID—the original family sequence number prior to the restructuring that is performed by the RecodeTeenHeads.sas code. ASECFamilyID is the same as FamilyID in households that are not restructured.

In summary, the following map records are used only in default conversions.

* AlienHouseholdSplit
* CorrectedCitizenship
* DetailedCorrectedCitizenship
* EditedCountryOfBirth
* EditedYearInUS
* HighIncomeClone
* ASECFamilyID

#### III.4. Review the CPS Technical Documentation and Update the CTD.Map and TRIM3 Code

Carefully review the March CPS ASEC documentation to identify any changes from the prior year, in particular the section titled “Differences Between the <prior year> and <current year> Annual Social and Economic Supplement Files.” Comparing record lengths for the three hierarchies (household, family, and person) with those of the prior year will also provide an indication of whether new variables have been added.

Also, be sure to check to see if there are **User Notes** at the end of the CPS ASEC Technical Documentation that identify any known problems with the data about which data users should be notified or that should be taken into account when converting the data.

##### Responding to Changes to the CPS

Invariably, the CPS input file changes over time. New variables are added and old ones are sometimes dropped, or new coding schemes are devised. Following is some brief guidance for responding to changes. Other sections of this document contain additional information.

* New variables are available. It is generally advisable to obtain the views of analysts who most often use TRIM3 data to determine whether to add new variables to the TRIM3 database. Not all CPS fields are of value to TRIM, and it may be unnecessary to enlarge the TRIM3 database for new fields. If a new field is added to the TRIM3 input database, create a new CTD.Map record for the field, assigning it a TRIM3 name and specifying map record values so Convert can read the data. Also add a dictionary entry for the new field in the CTD.Terms table, and add code values to the CTD.Enumerator table, linking the CTD.Terms and CTD.Enumerator entries with the Enumerator field.
* Old variables have been dropped. Typically, variables that have been dropped from the CPS are replaced by “filler” in the CPS documentation. Sometimes the old field has simply been repositioned on the data file so that it may accommodate a changed coding scheme or expanded range that requires more space. If a field is no longer available, search the TRIM3 simulation and frame code and the CTD.VariableList table to determine if the field is used by TRIM3. If so, then it may be necessary to create the discontinued field using other information that is available. If so, you will have to use a RecordType 4 record for the discontinued field in the CTD.Map table rather than RecordType 1-3 that was previously used.

For each variable that has been dropped that is a member of TRIM3 input schemas

1. Add a new class definition for the variable in *trim2.h,* using one of the existing classes as a prototype.
2. Implement the constructor and the member function *Calculate* for this class in *trim2.cpp*. Use the *Calculate* function to code the variable from other variables that are in the input data, or set the value to missing (-1000000000).
3. Register the class in the function *CGenManager::CreateGenClass* in *gen.cpp*.
* Making use of fields that are not added to the input database. You may need to reference a CPS field that is not added to the TRIM3 input schema in defining another field. If so, in the CTD.Map record, make the TRIM3 name the same as the CPS name to flag the fact that it will not be added to the TRIM3 database but is simply to be referenced by the Convert code. In the Convert code, define the field in convhset.h, register it in the Register function of the appropriate class, and add it to the LoadCreateCol function.
* A field has a new coding scheme. Since TRIM3 must be able to use historical data, field coding schemes must remain constant. Thus, if there is a change to the coding scheme of a field used by TRIM, it will probably be necessary to re-create the old field from the new one and possibly also add the recoded field as a new TRIM3 variable, since it is likely the new coding scheme makes available additional information. For an example of a similar adjustment made for changed information on race, see the TRIM3 variables “RaceAndSex” and “OldRaceAndSex,” and note the CTD.Map Type 4 record “COldRaceAndSex.” The coding of the “Race and Sex” TRIM3 field used extensively was kept unchanged, though changes in the Census race field resulted in some categorical differences from previous years. If this were not done, then it would be necessary to edit each simulation that uses the old field to accommodate both old and more recent input data.
* A field has a changed range of values and/or new location. Change the “RangeLow” and “RangeHigh” fields in the CTD.Map table and “MinimumValue” and “MaximumValue” in the CTD.Terms table to reflect any changed code range. If a field has been relocated, also change “Col” to the beginning column of the input record. If the length of a field has changed, update the misspelled “Lenght” field to the correct length. It is not necessary to update the maximum value for top-coded fields that change annually, but a note in the TRIM3 dictionary should explain that the field is a top-coded value whose maximum changes annually.

##### Peer Review of Coding Changes

Though sometimes unnecessary and at other times not possible, it is generally advisable to obtain peer review of any significant coding changes and to thoroughly test any changes made prior to the time they become routinely operable.

#### III.5. Update the CTD.Terms Table

The CTD.Terms table is the main data dictionary source table for TRIM3. It contains the names of all input fields, program rules, and results fields produced by TRIM3 simulations. Any new ASEC variables which are added to the TRIM3 input data must be defined in CTD.Terms before TRIM3 will be able to reference them. The terms table should also reflect the most recent definitions, codes, ASEC locations, and minimum and maximum values. It is not necessary to provide maximum values for top-coded fields whose maximum values are variable, i.e., that change annually. The CTD.Terms table contains the following fields.

|  |  |
| --- | --- |
| **Field** | **Description** |
| *Term* | TRIM3 variable name |
| *IsPublic* | A binary indicator (0 or 1) as to whether the data are accessible to members of the public. This is typically set to 1. |
| *Simulation* | Indicates the simulation that produces a result variable or for which a program rule variable is used. It is always set to “Input” for variables in input schemas such as the CPS. |
| *Sourcetable* | The name of the database table where the data reside. For input schemas, this is one of seven tables: Household, Family, Person, Adult, AdultCurrent, AdultPrevious, or AdultMonthly. |
| *Category* | Data category, generally one of the existing categories that have been defined for input data such as “1. Annual Income Variables” or “2. Detailed Income Components,” etc. |
| *Datatype* | “INT”, “REAL” and other data type specifications. This field is shown by the TRIM3 data dictionary but is otherwise insignificant. |
| *Oldname* | Indicates the old TRIM2 name for very old TRIM variables; it is null for all other variables. |
| *Description* | This is a comprehensive variable description that is shown by the data dictionary that includes notes and any other pertinent information a user of the TRIM3 data needs. HTML tags may be used to control formatting in the dictionary interface. Notes may indicate how codes or universes in past years differed from the most recent information. These notes are in lieu of a fully-functioning year-specific data dictionary. |
| *Enumerator* | The name of an enumerator that links information in the CTD.Enumerator table to the terms for which an enumerator is defined. |
| *MinimumValue* | Lowest valid value for a variable. |
| *MaximumValue* | Highest valid value for a variable. Not necessary for top-coded variables whose values change annually. |
| *MaximumNumberOfValues* | Used for array program rules. Not used for input schema variables. |
| *TrimUniverse* | Indicates the category of persons for whom a TRIM3 variable is coded—e.g., adults. |
| *CPSName* | CPS variable name. This is null if the variable is not on an input file. |
| *CPSLocation* | Location of the field on a CPS public release data file. The format for this information is a letter indicating hierarchy (H=Household, F=Family, and P=Person) followed by beginning and ending column positions (e.g., P32-34. The value is null if the variable is not on an input file. |
| *CPSUniverse* | Indicates the category of persons for whom a CPS variable is coded—e.g., adults. The value is null if the field is not on an input file. |
| *GroupName* | Used to categorize and group program rules; not used for Input terms. |
| *AggregateType* | Not used for Input terms. |
| *Subcategory* | A secondary category for grouping program rules; not used for Input terms. |
| *Sequence* | A sequence ordering number that controls the order in which variables are displayed by the TRIM3 interface. |
| *ValueEnumerator* | Similar to Enumerator. The name of an enumerator that links a term in the CTD.Terms table to code values in the CTD.Enumerator table. |
| *UsedInSimulation* | Intended to identify variables that are used by simulations, but not reliably coded. |
| *VersionFrom* | Indicates the version of a simulation that first requires use of a program rule term. Not used for Input terms. |
| *VersionTo* | The last version of a simulation that requires use of a program rule term. Not used for Input terms. |

**III.6. Update the TRIM3 TrimDLL and Convert Projects for the Current Data Year**

# Annual Updates

* Compute the current data year’s value for the CPI array using the Consumer Price Index value obtained in step II.4. Values in the CPI array are calculated by dividing the CPI for the current data year by the CPI for the base year (1967). For example, if the CPI for 1991 is 136.2 and the CPI for 1967 is 33.4, then the value enterend into the CPI array for 1991 is 136.2 / 33.4 = 4.078. Update the value of CPIsLastYear in Commun.cpp with the current data year, add the computed CPI value for the current data year to the CPI array, and increment the CPI array index in both Commun.cpp and Commun.h by 1.
* Update the BlsEstimatedPersonsWorking and BlsEstimatePersonsUnemploy arrays in Convert/Allocate.cpp using the employment and unemployment statistics obtained in step II.5 and increment the first array index by 1.
* Update the maximum data year permitted in the MonthlyAllocation function in Convert/Allocate.cpp.
* Update the PassThruChildSupport array in commun.cpp for the new year and increment the first index by 1. Also update the value of LastYearPassThruUpdate to indicate the last year for which data are available.
* Update the WeeksPerMonth array in WeeksPerMonth.cpp for the new year and increment the first index by 1.
* Update the maximum data year permitted in the CalculateFactors function in Factor.cpp.

# Other Possible Updates

If new variables are contemplated, consult with TRIM3 management personnel to determine if the data will simply be added to the results schema in a supplementary table that can be accessed by simulations via variable list program rules, or will be incorporated into the TrimDLL project, commonly referred to as the frame. Currently the frame does not have the capability to access new fields that are added to the standard input schema tables via the Convert project but that are not also added to the TrimDLL project. Incorporating new fields into the frame (or deleting existing fields), however, has broad implications. To do so requires that all input schemas be expanded (if new fields are added), all Make files re-created, and all simulations recompiled and re-verified. This is time-consuming work for which there is typically insufficient time or budget.

If Convert must create a new variable that is not directly input, create a new class whose coding is invoked by a Type 4 CTD.Map record. To do so, search the Convert code for an example to follow, such as OldRaceAndSex. Also refer to the Section III.4 discussion of dropped fields for information on generating a variable that is not on an input file.

Any new fields added to the Convert project must be declared in Convert/convhset.h and added to the Register and WriteRec functions of the class of which they are a part in Convert/Houseset.cpp. Any new fields added to the TrimDLL project must be declared in TrimDLL/hset.h and added to the FetchRec and Register functions of the class of which they are a part in TrimDLL/Houseset.cpp.

**III.7. Perform SAS Pre-processing of the Input Data**

Obtain the SAS code referenced here from the lead TRIM3 programmer, or copy the latest versions of the code from the source folder of the ASPE-TRIM application server. All SAS code referenced here must be edited as needed for the current data year. Edits may be as simple as updating year. Confer with the lead TRIM3 programmer and see code comments for further assistance.

The first SAS code that must be run for any conversion of CPS ASEC data is RecodeTeenHeads.sas. It alters the structure of “inverted households,” i.e., households in which a teenager or young unmarried adult living with at least one parent has been identified as the householder.

If you are performing an alternative conversion that does not clone high income households or merge in immigrant imputations, no other SAS codes must be run before running the TRIM3 convert process. In that case, the output file created by RecodeTeenHeads.sas is used as the input file for the TRIM3 convert process.

However, if you are doing a default conversion that requires cloning households and merging in immigrant imputations, then use the output file created by RecodeTeenHeads.sas as input to the remaining SAS code that must be run for a default conversion. To clone high-income households for the SOI match, run IdentifyHighIncomeHhs.sas first, followed by Clone.sas. Before running IdentifyHighIncomeHhs.sas, ascertain that the income values it uses in identifying high-income households are the ones applicable to the file you are converting by checking the ASEC documentation accessible from the Census Bureau web site at <http://www.census.gov/programs-surveys/cps/technical-documentation/complete.html>.

Run CheckImmigrantData.sas to check the immigrant imputations. The code generates statistics that may be examined and ascertains that the sum of person weights read from CPS ASEC data agree with the sum of the split weights from the imputed data. It also provides a test of the unique person identifiers used to merge records in the ImmigrantMerge SAS code, and it provides tabulations of immigrant status by age that you may compare with post-conversion database values. The immigrant status by age tabulations produced by CheckImmigrantData.sas may not exactly match ones produced by person or center who provides the imputations since age may be edited during the imputation process.

You may then run the final SAS code—ImmigrantMerge.sas—which reads the output files created by RecodeTeenHouseholds.sas, Clone.sas and CheckImmigrantData.sas and produces an output file that is used for the TRIM3 Convert process.

**IV. Run the Conversion**

## IV.1 Run Convert via the TRIM3 Interface

Re-compile Convert in release mode after performing the required updates and copy the resulting Convert.dll to the server that is to be used for conversion. Copy the input data file that is created by the applicable pre-processing SAS code(s) to a directory that is accessible to the TRIM3 server. Enter into the TRIM3 console window the correct name and path of the input file. (If TRIM is running as a service, the console window will not be accessible, so you may have to either run Convert locally or temporarily disable the service and run TRIM3 interactively.)

Using the TRIM3 web interface, go to the Simulation Runs page, select the server to be used for conversion, and click on “Convert/Make.” Select the appropriate year using the pull-down box, and check the Convert box, then press O.K. If the interface fails to show the appropriate year, then check to ascertain you entered a new record into the CTD.InputSchemas table for the new data year and that the information is correct. It is not necessary to leave the TRIM3 dialog box open once Convert is launched.

Conversion typically requires less than two hours to complete, depending on server speed, input file size, database connection speed, and database activity. Check server status to determine when Convert has finished.

##### **IV.2 Conversion Data-flow Diagram**



**V. Verify Conversion and Create a New TRIM3 Input File**

##### **V.1 Perform an Initial Check of the Data**

As data are being converted, you may immediately begin scanning the new input schema tables for possible errors or oversights. The AdultMonthly table will only be populated once all input records are read and other tables have been completely populated. If you spot an error that requires correction and a repeat of conversion, you may immediately stop Convert, delete all input tables (but not the schema), correct the error and restart Convert. Use MySQL commands such as the following to drop the conversion tables so that you may repeat conversion.

Drop Table CPSyyyy.Household;

Drop Table CPSyyyy.Family;

Drop Table CPSyyyy.Person;

Drop Table CPSyyyy.Adult;

Drop Table CPSyyyy.AdultCurrent;

Drop Table CPSyyyy.AdultPrevious;

Drop Table CPSyyyy.AdultMonthly;

##### **V.2. Review the Convert Log**

Once Convert completes successfully, review Convert.log, which is located in the server directory that is specified by Convert project settings. Carefully check log messages for any indication of problems. Check record counts and ascertain they are consistent with the CPS documentation, with input schema table counts and with counts from the SAS log of the code that produced the Convert input file. This log contains some overall statistics as well as a record of variables which triggered errors or warnings.

The statistics include:

1. Total CPS Households.
2. Total TRIM Households.
3. Households deleted because they were not interviewed. This should be the difference between CPS and TRIM households.
4. Total CPS Families.
5. Total TRIM Families.
6. Child-only families merged with another family. This should be the difference between CPS and TRIM families. There should be approximately 300 of these.
7. Total CPS Persons.
8. Total TRIM Persons. This should be the same as CPS persons.

Make sure out-of-range messages are not indicative of significant errors that must be corrected. You may simply need to update the RangeLow and/or RangeHigh Map fields to prevent future messages. Out-of-date RangeLow and RangeHigh values in the CTD.Map table will not impose limits on data that are read in. Their only value is in helping programmers spot errors via the Convert.log. (If you update RangeLow and RangeHigh values in CTD.Map, also update MinimumValue and MaximumValue in CTD.Terms for the same terms.)

Out-of-range messages can, however, indicate that a field’s location and/or width or coding scheme has changed and that a corresponding change was not made to the CTD.Map record. If a field is read in incorrectly, the Map table must be corrected and Convert must be repeated.

# The following table provides information about possible error or warning messages you may see in the in the log file.

|  |  |
| --- | --- |
| *The BeginIndex for <Variable> is too big.* | A variable’s beginning column defined in CPS.Map exceeds the CPS record’s length. |
| *The conversion was aborted* | You or another person or process cancelled the conversion. |
| *The following columns in <table> were duplicated* | The variables listed have more than one mapping. Repeat the conversion, removing these duplicates from CTD.Map. |
| *The following columns in <table> were not used* | The columns in the CPS data record do not contain the value of any defined variable. Since the CPS format contains filler characters, this is not necessarily an error. |
| *The following signals in the Input database were not generated* | Check to ascertain that none of the listed variables are needed. If they are, the map table must be corrected and conversion repeated. |
| *The following signals in the Map table are not used by a CGen class* | This message refers to *RecordType* 4 CTD.Map records that are not actually used in any of the conversion calculations. |
| *The following signals in the Map table have problems* | CTD.Map entries for these variables contain missing or invalid information. Such messages may indicate that the CTD.Map table must be corrected and conversion repeated. |
| *The Map Table has the wrong record type.* | *RecordType* in CTD.Map has an invalid value. |
| *<Variable> has an index of zero, no value is assigned* | The variable’s starting column (offset) in CTD.Map is zero. Since the first position in the CPS record is one, the variable’s value is undefined. |
| *<Variable> is in the Map table but is not used.* | The variable’s starting column in CTD.Map is zero. |
| *<Variable> is outside the range <n> times* | Values of the specified variable do not fit the range specified in the variable mapping.  |
| *<Variable> the Rec Type does not agree with the Var Rec Type* | *RecordType* in the map table is not compatible with the TRIM3 table name. For example, a *RecordType* 3 (Person-level) variable should not be placed in the Household table. |
| *The Map Table is empty.* | The year you specified in ‘*Simulating data’* does not have a variable mapping. Set the year to one year less than the survey year which you want to convert. |
| *The CPS file cannot be opened.* | The CPS file path on the simulation engine window is invalid or the input file has not been copied to the path shown in the Console window. |

# **V.3. Perform Required Manual Database Edits**

Manual database edits may be required for immigrant conversions, for instance to correct cases where sample persons are reported to be natives born in the U.S., but country of birth is outside the U.S. The person who supplies the immigrant imputations data should also supply you with any required manual edits to the TRIM3 database, typically just a small number of edits to the Person table that may be made using a MySQL tool such as Browser. Such edits are not always necessary.

#####

##### **V.4.** **Make the New Schema Accessible to Registered TRIM3 Users**

To make the new schema accessible to registered TRIM3 users, run the trim\_verify\_user\_accounts Perl script by doing the following:

* Log onto the application server with your username and password and open a command window.
* Making use of environmental variables for access to the database servers, type the following command: “perl d:\source\scripts\trim\_verify\_user\_accounts.pl -u %trim\_db\_user% -p %trim\_db\_pass% -h %trim\_db\_server% -P %trim\_db\_port%”

[NOTE: appending “-oUserName” will cause the script to execute for the single user whose user name is provided.]

Notice that the password argument is lower case, whereas the port argument is upper case.

##### **V.5. Refresh the Database from the Application Server**

To refresh the ASPE-TRIM database, log onto application server that accesses the database server where the new schema exists and set the default directory to d:\inetpub\wwwroot\_prod\microdata. Open a command prompt window and type “refresh” to execute refresh.bat that resides at the above location on each application server. Keep the command window open until the refresh process completes; it may require several minutes. Error messages, if any, will be displayed.

#

# **V.6. Generate Term Statistics**

Generate term statistics for a new schema using the TRIM3 Perl script that generates the following distribution statistics for each variable in a year’s input data:

* Minimum value
* Maximum value
* Number of zero values, weighted and unweighted
* Number of non-zero values, weighted and unweighted
* Sum of values, weighted and unweighted
* Mean value, weighted and unweighted
* Mean of non-zero values, weighted and unweighted

Generating term statistics for a new schema will enable you to compare new schema fields with a comparable schema to spot unexpected differences that may alert you to possible conversion problems. To generate term statistics, log onto a TRIM3 application server and open a command window. (The default directory is insignificant.) Then execute the following command:

“trim\_input\_stats.pl -u %trim\_db\_user% -p %trim\_db\_pass% -h %trim\_db\_server% -P %trim\_db\_port% -i SchemaName”. Notice that the password argument is lower case, whereas the port argument is upper case. If an “InputStats” batch file exists (expectedly in the root directory of the D drive), you may simply edit and run the file for the schema for which you want to generate statistics.

NOTE re trim\_input\_stats.pl: This Perl script not only generates term statistics, it also populates the table ctd.InputVarYears, which must contain an entry for every term in an input schema, with the exception of fields having all null or zero values. (An exception is made for the person-level “variable,” ZeroField, which must be made available for selection in run setups and which contains all zeros.) ctd.InputVarYears controls the variables a user is able to access during simulation set-up. If a variable exists but has no entry in that table, it will not show up in the TRIM3 interface. Thus, if a variable is added to an input schema following conversion, then this script should be re-run to re-generate term statistics and to re-populate the ctd.InputVarYears table for the affected schema(s).

Keep the command window open until processing is finished. This process may take up to an hour to complete. Once statistics are generated, close the command window and log off the server. Then, using a MySQL GUI tool, connect to the TRIM3 database on Cybele and issue all of the applicable MySQL commands in <CompareTermStatistics.txt> to identify any significant changes to input schema variables. You will need to customize the commands in the text file to identify the newly converted schema and the schema with which you are comparing, and you may change the parameters that control the magnitude of difference you want to identify. Some of the SQL queries are designed to compare a newly converted schema with one from the prior year, and others compare an immigrant schema with the initial one using the same input data. Because of problems with two terms in prior conversions—DetailedOtherIncome and GeneralOtherIncome—there are two queries that individually check these terms. Comments in the text file may be useful in identifying query results you should expect. Carefully check query results for any indication of conversion problems that need to be corrected or significant variable changes about which analysts need to be alerted.

# **V.7. If Required, Correct the Microdata.Variable Table for the Current Year**

If the refresh procedure is run but the Make process fails, check the Microdata.Variable table to ascertain that it contains all the variables in the newly converted data. The Microdata table must contain a record for every variable from the newly converted schema that is written to the Make file. If it does not, the Make process will fail. If it is necessary to add records manually, refer to the folloing paragraph. Otherwise, this step may be skipped. (Note that there are some variables that are written to database tables that are not written to the Make file; records do not have to be created for these variables in the Microdata table, though it does no harm if they are created by the refresh procedure.)

The easiest way to make sure the Microdata.Variable table contains all of the correct records is to delete all records for a new schema that were automatically added by the refresh procedure and then copy a standard set of records from another table. To delete existing records, use MySQL commands of the form “Delete from microdata.variable where form = ‘2004 Household’.” You will need to issue such a command for each of the seven input schema tables. The Microdata “form” field is derived from the “Year” field of CTD.InputSchemas, with table name appended, thus for immigrant imputations, the delete command for the household table would be “Delete from microdata.variable where form = ‘Alien2004 Household’.” Once the records that were automatically added have been deleted, edit the form field in Microdata.std\_variable\_set for the new schema. Then copy all records from Microdata.std\_variable\_set to Microdata.Variable. [EditMicrodataTable](file:///%5C%5CChiron%5CC%24%5CInetpub_IIS%5Cwwwroot%5CDocumentation%5CTechnical%5CEditMicrodataTable.doc) may be of help with this step.

# **V.8. Generate and Test a New Make File**

To read a large volume of input data quickly, the TRIM3 frame stores it in a file, called a ‘Make’ file. Once term statistics compare as expected and there are no other indications of conversion errors, then create a Make file. Because of annual updates made to commun.h and commun.cpp, you need to first recompile the TrimDLL project in release mode and copy TrimDLL.dll to the server that will be used to create the new Make file. [If future changes are made to incorporate these annual data into database tables(s), then it will no longer be necessary to recompile the frame code every year when updates are made.]

Run Make via the TRIM3 interface. Select server and then click on Convert/Make. In the pop-up window, select data year, click Make, then click O.K. Since the Make process prevents other TRIM3 processes from executing while it is running, you may want to run Make locally so that other users may continue to use the public server.

After a Make file has been created, the TRIM3 engine must be stopped to release the file handle. Otherwise, efforts to copy, move, or access the newly-created Make file by simulation code may result in an “access violation” error. This problem may be corrected in later releases of the C++ development software.

After creating a Make file and releasing the file handle, test the new input file in a run of the ChildCareExp simulation. Create a setup for the new schema by copying the “CCE\_VerifyConvert” setup [under the name “JMORTON” and the 2006 Alternative (no immigrant imputation) schema]. The setup needs no change other than input schema. It should complete without error.

**VI. Post Conversion Tasks**

Conversion is not complete until the post-conversion tasks have been completed. These include the following steps.

* Refresh the database a second time as instructed in step V.5. to repopulate the Microdata.Variables and CTD.InputVarYears tables.
* Save any changed C++ code to Cybele/source and any re-compiled Convert.dll to Cybele C:/Trim3Svr, either keeping or replacing existing code according to current practice.
* If using CVS or some other version control repository, commit the updated code to the repository with an appropriate name, e.g., “Convert\_ASEC\_yyyy” (where yyyy designates survey year) or “Convert\_ASEC\_yyyAlien” (for immigrant conversions).
* Update the “What’s New” page in the Cybele interface to inform users of the availability of the new input data.
* Create new CYYear\source and CYYear\Imputed Immigrant Data folders on Cybele\Source in the “Convert, SAS Source and Data” folder. Save any updated SAS codes used in conversion to the source folder, and copy the immigrant imputation data to the data folder.

**VII. Conversion Process Diagram**



**VIII. Appendix A: Procedure for Adding a New CPS VariableTo Input Schema Tables Only**

It is possible to add new CPS variables to an input schema but not to the TRIM3 frame. Such variables may be accessed by the TRIM3 microdata tool for extracts and may be specified as inputs in variable list rules, but they are not written to the Make file and cannot be accessed directly by simulations. Following is a summary of the steps required for adding new input variables. These instructions are intended to supplement the information contained in the body of this document, not to provide the comprehensive information that is available therein.

1. Add the new variable to the TRIM3 dictionary (the ctd.terms table). Consider not renaming the variable for TRIM but, instead, retaining the CPS name. This practice can help distinguish between variables that are on the Make file and ones that are not.
2. Add the mapping details for the new variable to ctd.map for the input schema to be converted.
3. Add the new variable to the TRIM3 Convert project only, not to TRIMDLL. To do so:
* add the variable to the appropriate structure in convhset.h depending on hierarchy—i.e., household, family, person, adult or adultmonthly;
* register the variable in houseset.cpp in the appropriate class;
* add it to the WriteRec function of the appropriate class.
1. Recompile the Convert project and perform conversion for the new schema following the usual conversion procedures.
2. If the new variable is to become a part of later schemas, add a record for it to microdata.std\_variable\_set prior to copying those records to microdata.variable. If the new variable is to be a part of a single schema, then add a record directly to microdata.variable for the single schema. Then follow the standard procedure for creating a Make file.
3. Once Convert and Make have completed, ascertain that the set of records in ctd.InputVarYears for the new schema includes the new CPS variable.
4. If the new CPS variable is not to be included in later schemas, then reverse Step 3 in the Convert code.
1. This table was formerly identified as “Consumer Price Index, All Urban Consumers, Current Series, the most requested statistics, U.S., All Items with 1982-84 =100, Not Seasonally Adjusted.” [↑](#footnote-ref-1)
2. Inverted households are households in which an unmarried teenager, young-adult student, or young adult with a disability is identified as the householder, even though (s)he lives with one or both parents. [↑](#footnote-ref-2)