SENTINEL TOOLKIT

COMBO TOOL TO DEFINE EVENTS OF INTEREST USING COMPLEX ALGORITHMS

Prepared by the Sentinel Operations Center
June 28, 2016
Version 2.6

Sentinel is sponsored by the U.S. Food and Drug Administration (FDA) to monitor the safety of FDA-regulated medical products. Sentinel is one piece of the Sentinel Initiative, a multi-faceted effort by the FDA to develop a national electronic system that complements previously existing methods of safety surveillance. Sentinel Collaborators include Data and Academic Partners that provide access to health care data and ongoing scientific, technical, methodological, and organizational expertise. The Sentinel Coordinating Center is funded by the FDA through the Department of Health and Human Services (HHS) Contract number HHSF223201400030I. This project was funded by the FDA through HHS Mini-Sentinel contract number HHSF223200910006I.

Table of Contents

Modification History

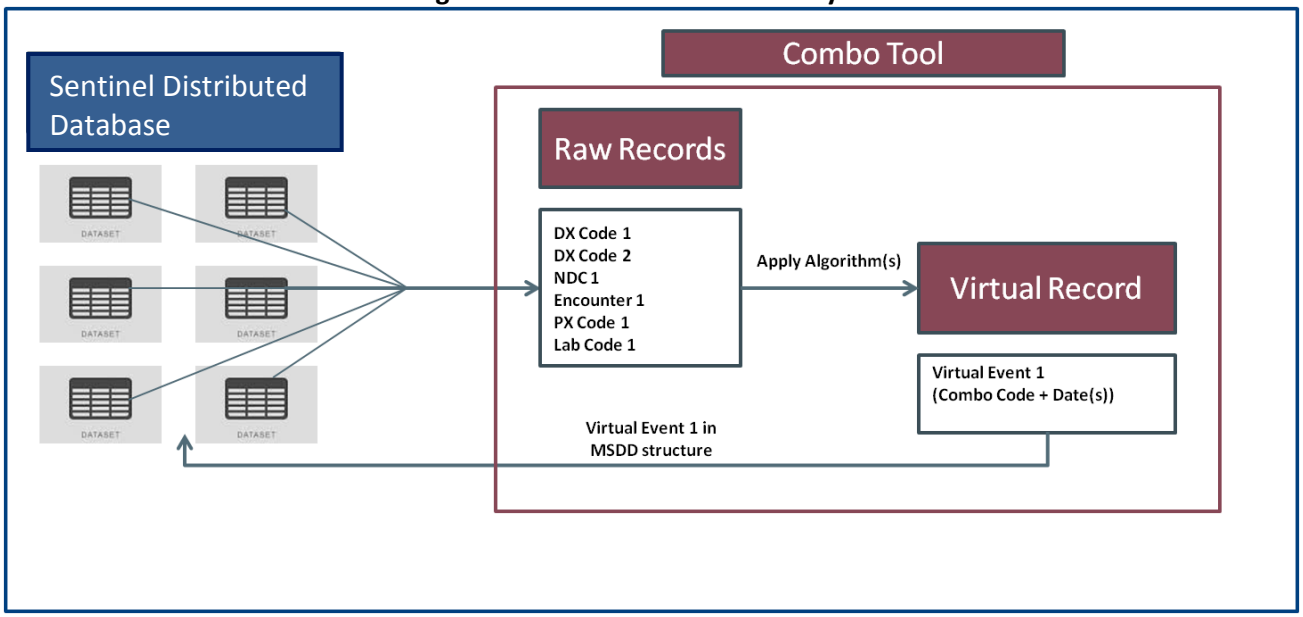
Version	Date	Modification	By
2.6	04/15/2016	• Updated lab code map lookup file	Sentinel Operations Center
2.5	09/29/2015	• Modified to allow processing of demographic criteria	Sentinel Operations Center
2.4	09/22/2015	• Changed default for PERCENTDAYS from 0 to missing	Sentinel Operations Center
2.3	06/26/2015	• Minor bug fixes	Sentinel Operations Center
2.2	02/23/2015	• Minor bug fixes	Sentinel Operations Center
2.1	01/12/2015	• Minor bug fixes	Sentinel Operations Center
2.0	09/15/2014	• Modified code to allow processing of combinations of combinations. New parameter, "CombOrder", added to Combo Input File	Sentinel Operations Center
1.4.2	07/10/2014	• Minor bug fixes	Sentinel Operations Center
1.4.1	05/27/2014	• Minor bug fixes	Sentinel Operations Center
1.4	05/09/2014	• Included new parameter "RAWDURAT" in Combo File, allowing user to set a fixed duration of follow-up following the occurrence of a procedure code • Minor bug fixes	Sentinel Operations Center
1.3	05/02/2014	• Updated based on QC suggestions following V1.0 to V1.2 audit	Sentinel Operations Center
1.2	01/06/2014	• Updated inputs based on Dec 2013 comments on specifications	Sentinel Operations Center
1.1	07/31/2013	• Addressed Comments and update	Sentinel Operations Center
1.0	07/02/2013	• Initial Draft	Sentinel Operations Center

I. OVERVIEW

Many algorithms to identify exposures and outcomes of interest in electronic health data require identification of a combination of items to define a single, valid event. The primary purpose of the Combo Tool is to have a re-usable SAS macro that can implement a wide array of complex algorithms, and reduce the need for de novo programming each time a new algorithm is developed as part of a Sentinel workgroup or routine data query.

The process of creating single events from combinations of multiple Sentinel Distributed Database (SDD) variables and variable values is simple and flexible. The SAS macro processes multiple "raw" items used to determine a combination and creates a single "virtual" record that summarizes the combination in a fashion identical to raw SDD diagnosis, procedure, or dispensing records. Users of the tool determine how the final virtual record should *behave*; *i.e.,* if the record should mimic a diagnosis, procedure, or dispensing record, and what date(s) virtual records are assigned. Figure 1 summarizes how the Combo Tool creates virtual events.

Figure 1: Combo Tool Functionality



The Combo Tool can perform the following functions:

- Combine any NDCs, diagnoses, procedures, encounter types, enrollment episodes, lab values, and demographic criteria (using "and" and "or" joins)
- Use same-day, same-encounter, or time intervals to define events (*e.g.,* diagnoses X and procedure Y within 2 weeks of each other)
- Let the user decide what "event date" is used on the virtual record for combinations that are defined with spans of more than one day
- Define a specific exposure length for any codes that comprise the combination (similar to days of supply or length of stay)
- Let the user have the ability to define the encounter type for the resulting virtual record

A. COMBO TOOL USE

The Combo Tool is a standalone SAS macro designed to be easily 1) used as a standalone tool by SAS programmers writing new code to query the SDD; and 2) integrated into existing Sentinel routine query tools (*e.g.,* modular programs). While the Combo Tool is functionally equivalent regardless of how it is used, there are some considerations for standalone versus integrated use:

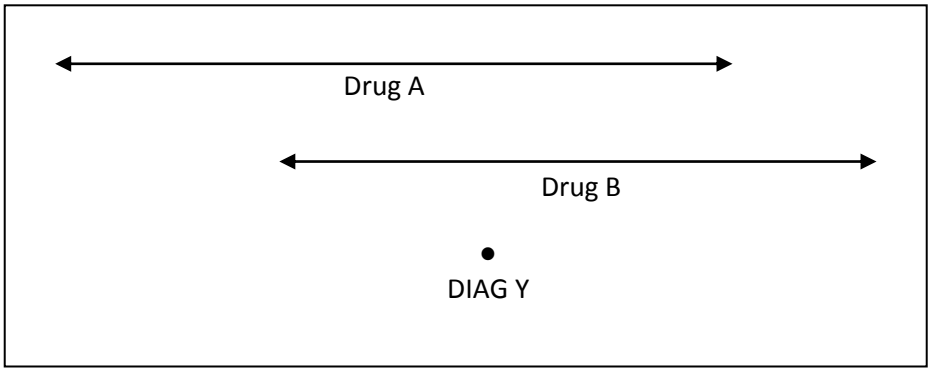
- Macro parameters: due to how the Combo Tool is integrated with the modular programs, some macro parameters differ between standalone and integrated implementation of the tool. This documentation describes the macro parameters that must be specified for the successful execution of the standalone tool; reference the modular program documentation for information on macro parameters that must be specified for integrated use.
- Eligibility requirements: modular programs allow for the specification of enrollment eligibility requirements – by default, all codes identified must occur during a valid enrollment span. Additionally, users can define the required number of days of continuous enrollment before an index date to determine eligibility for cohort inclusion. The standalone Combo Tool, however, does not have a straightforward method to specify either of these requirements. As such, programmers using the standalone tool **must specify and ensure eligibility requirements as part of their program**.

II. STEPWISE TWO-BY-TWO METHOD

Combinations are created using a *stepwise two-by-two* method. This method consists of using the first two raw items to create an "intermediate" virtual combination record, and then combining the intermediate virtual record with the next raw combination item. This process is repeated until all combination items have been processed. The final virtual record will meet all the combination definition requirements. The final virtual record is then used "as-is" by programs designed to query the SDD. To better illustrate the method, consider the following combination scenario:

Suppose we need to identify subjects taking the combination therapy Drug A and Drug B and that have had an outcome of diagnosis DIAG Y over the course of the combination therapy. Figure 2 illustrates a valid case of the scenario:

Figure 2: Scenario Illustration

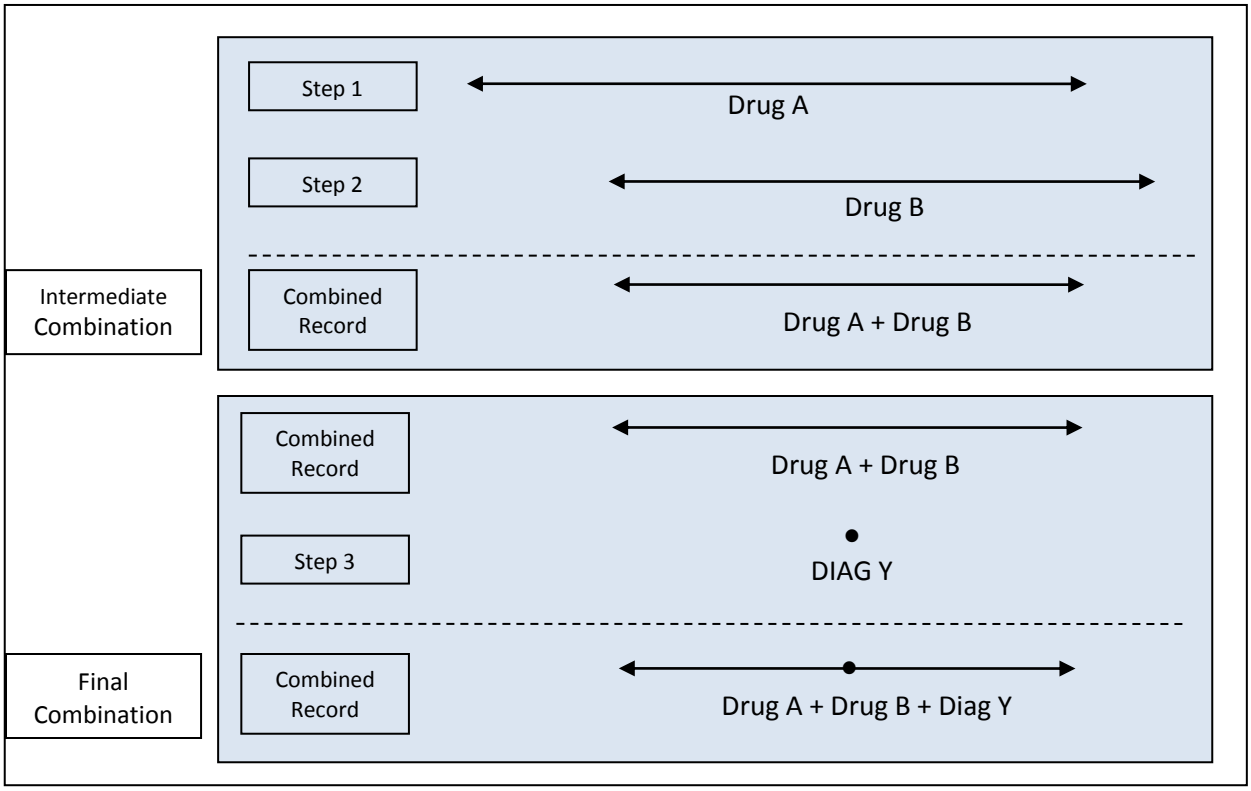


The ordered steps involved in creating the combination item include:

Step 1: Identify Drug A treatment
Step 2: Identify Drug B treatment overlapping Drug A treatment; create virtual combination record
Step 3: Identify Diagnosis Y codes that overlap Step 2 virtual combination record

The stepwise two-by-two method will first process Steps 1 and 2 and create an *intermediate* virtual combination record. Then this intermediate virtual record will be used to complete Step 3. Figure 3 illustrates the algorithm.

Figure 3: Stepwise Two-by-Two Method



A. COMBINING ITEMS

In the SDD, there are essentially two types of "raw" items: one day events (*e.g.,* diagnoses, procedures); and multiple day events (*e.g.,* dispensing with days of supply > 1, inpatient stays, etc.). With these two types of raw items, a total of three types of combinations can be created:

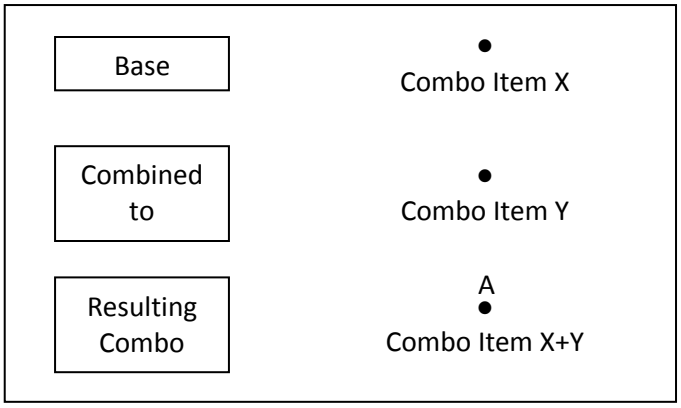
1. One Day <-> One Day
2. One Day <-> Multiple Days
3. Multiple Days <-> Multiple Days

1. One Day <-> One Day Combinations

This type of combination is the most straightforward. To be included in this category, records for the two sets of combination items must occur on the same day. The typical use for this type of combination

is to combine diagnoses and procedures. For example, the user may want to identify patients with a depression diagnosis, but without a schizophrenia diagnosis on the same day, or identify patients with a diagnosis for hypertension coupled with a CT scan procedure on the same day. Figure 4 illustrates a one day <-> one day combination where the algorithm is identifying single events where X and Y occur on the same day.

Figure 4: One Day <-> One Day Combinations



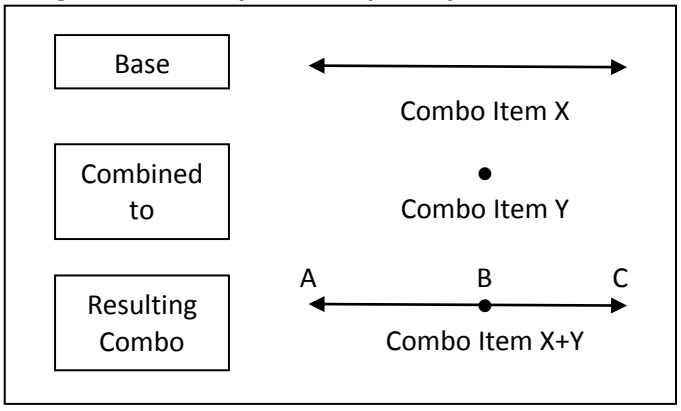
In this case, there's only one possibility for the virtual record event date, which is the item X and item Y date (A).

2. One Day <-> Multiple Days Combinations

To be included in this category, one of the two combination items must be a single day event. The typical use for this type of combination is to combine diagnoses and procedures with drug use, inpatient stays, or eligibility episodes. For example, the user may want to identify members with liver transplant during a hospital stay or with a diagnosis for arrhythmia during the course of drug treatment.

Figure 5 illustrates a one day <-> multiple day combination where the algorithm is combining multiple day item X with single date item Y.

Figure 5: One Day <-> Multiple Days Combinations



Contrary to the same day event category, one day <-> multiple days combinations have up to three choices for start date and end date for the final virtual record (A,B, or C).

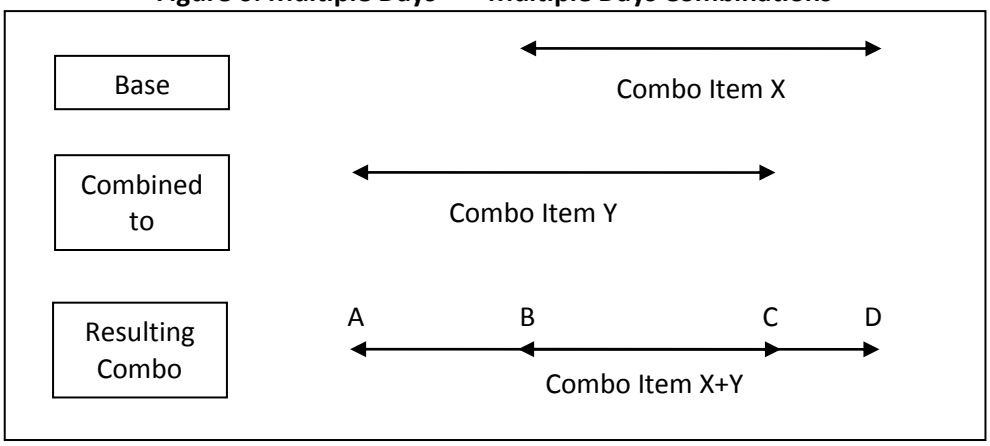
3. Multiple Days <-> Multiple Days Combinations

This is the last but most general type of combination. To be included in this category, both combination items must span more than one day.

The typical use for this type of combination is to combine several drug uses (combination therapy) and/or or inpatient stays and/or eligibility spans. For example, the user may want to identify members with a hospital stay with evidence of at least two days of supply for a drug.

Figure 6 illustrates a multiple day <-> multiple day combination where the algorithm is combining multiple day item X with multiple day item Y.

Figure 6: Multiple Days <-> Multiple Days Combinations



For this category, we now have up to four choices for start date and end date for the final virtual record (A, B, C, or D).

B. DEFINING VIRTUAL RECORD DATE(S)

Because the virtual record can be the result of combining intervals, there are many possibilities to define start dates and end dates for intermediate and final virtual records.

For each intermediate or final combination processed (*i.e.,* each time two items are combined), we define a "Base" record and a "Combined to" record. The Base record always represents either the first or an intermediate combination item, while the "Combined to" record always represents a raw item. Each time two combination items are processed, the user must determine which start and end date to retain for the combined record. If the Base + Combined to combination is an intermediate combination, the dates selected will be carried forward for additional combination processing. If the combination is a final combination, the start and end dates will determine the dates used for the final virtual record. The following start and end date options are available:

- The start date of the Base interval
- The start date of the Combined-to interval
- The minimum date between the Base and Combined-to start date
- The maximum date between the Base and Combined-to start date

- The end date of the Base interval
- The end date of the Combined-to interval
- The minimum date between the Base and Combined-to end date
- The maximum date between the Base and Combined-to end date
- The start of the overlapping period between Base and Combined-to interval
- The end of the overlapping period between Base and Combined-to interval

C. VIRTUAL RECORD LAYOUT

Once dates are specified, the last step is to tell the Combo Tool how the final virtual record summarizing the combination should behave. This is important because then the virtual combination record can then be used by any programs designed to query the SDD (*i.e.,* final virtual records can be queried just like any other record in the SDD). Final virtual records can ^ behave like Diagnosis, Procedure, or Dispensing table records in the SDD. See Section IV.

III. COMBO TOOL PARAMETERS

A. STANDALONE COMBO TOOL MACRO PARAMETERS

Several macro parameters can be specified. These include input file names, envelope indicators, a stockpiling indicator, enrollment gap used to create continuous enrollment periods, and output file names.

Note that the macro parameters specified here are those associated with the standalone Combo Tool. Macro parameters associated with the modular program integrated Combo Tool are different and are described in the modular program documentation.

Table 1: Combo Tool Macro Parameter Specification

Parameter	Field Name	Description
Combo Input File	COMBFILE	**Details:** Name of the SAS dataset defining the Combo Input File. **Named by:** Request programmer **Input type:** Required **Format: .**sas7bdat **Example:** COMBFILE=Hypoglycemia
Combo Codes Input File	CODESFILE	**Details:** Name of the SAS dataset defining the Combo Codes Input File. **Named by:** Request programmer **Input type:** Required **Format: .**sas7bdat **Example:** CODESFILE=HypoglycemiaCodes
Envelope Indicator for Diagnoses	DXENVEL	**Details:** Indicator to determine if the envelope should be run on diagnosis claims. If DXENVEL=1 the envelope will be run, otherwise it will not. Default value is empty, meaning the envelope will not run.

Parameter	Field Name	Description
		Note 1: The envelope reclassifies all diagnoses associated with outpatient and emergency department encounters as inpatient if the encounter overlaps an inpatient stay. **Named by:** Request programmer **Input type:** Required **Format:** Numeric **Example:** DXENVEL=1
Envelope Indicator for Procedures	PXENVEL	**Details:** Indicator to determine if the envelope should be run on procedure claims. If PXENVEL=1 the envelope will be run, otherwise it will not. Default value is empty, meaning the envelope will not run. **Note 1:** The envelop reclassifies all procedures associated with outpatient and emergency department encounters as inpatient if the encounter overlaps an inpatient stay. **Named by:** Request programmer **Input type:** Required **Format:** Numeric **Example:** PXENVEL=1
Output File	OUTNAME	**Details:** Name of the prefix that will be used to produce the output files in DPLocal if the SAVETODPLOCAL parameter is set to 'Y'. For more information about output files, please refer to Section IV below. **Named by:** Request programmer **Input type:** Required **Format:** Alphanumeric **Example:** OUTNAME=ComboOutput
Stockpiling Indicator	STOCKPILING	**Details:** Indicator to determine if stockpiling should be executed on outpatient pharmacy dispensings. If STOCKPILING=1 the stockpiling algorithm will be executed. Default value is empty, meaning the stockpiling will not be executed. **Note:** When STOCKPILING=1, the STOCKPILINGFILE parameter must be specified. **Named by:** Request programmer **Input type:** Required

Parameter	Field Name	Description
		Format: Numeric **Example:** STOCKPILING=1
Combo Codes Stockpiling Input File	STOCKPILINGFILE	**Details:** Name of the SAS dataset defining the stockpiling settings for each RawGroup found in the Combo Codes Input File. **Named by:** Request programmer **Input type:** Optional if the STOCKPILING parameter is not equal to 1 **Format: .**sas7bdat **Example:** STOCKPILINGFILE =HypoglycemiaStock
Enrollment Gap	ENROLGAP	**Details:** Sets the number of days that will be bridged between two consecutive enrollment periods to create a "continuously enrolled" period. For example, if ENROLGAP=30 and a member is eligible for medical and drug coverage in periods 1/1/2007-3/27/2007 and 4/1/2007-12/21/2007 (*i.e.,* a 4-day gap between two consecutive enrollment episodes), the member will be considered continuously enrolled from 1/1/2007 to 12/21/2007. Any gaps in enrollment greater than 30 days will result in a new enrollment period, and all the days in the gap will be considered un-enrolled. **Note 1**: A gap of 45-days is recommended for most uses. **Note 2:** Multiple continuous enrollment periods per member may be assessed. **Note 3:** If left empty, enrollment episodes will not be bridged. **Defined by:** Requester **Input type:** Required **Format**: Numeric **Example**: ENROLGAP=45 (gaps less than or equal to 45 days will be "bridged" to form one "continuously enrolled" sequence)
Lab Code Map File	LABSCODEMAP	**Details:** Name of the SAS dataset defining the Lab Code Map File. **Named by:** Request programmer **Input type:** Optional **Format: .**sas7bdat **Example:** LABSCODEMAP =Lab_Lookup

Parameter	Field Name	Description
Output to DPLocal Indicator	SAVETODPLOCAL	**Details:** Indicator to determine if output files should be created in DPLocal. If this parameter is empty or has a value of 'N' the output files will not be created. **Named by:** Request programmer **Input type:** Optional **Format:** Alphanumeric **Example:** SAVETODPLOCAL=Y

B. COMBINATION DEFINING INPUT FILES

Two input files specify how combination items should be created. Although numerous variables can be required to define complex combinations, they can be divided in two categories for a better understanding of their role in the algorithm: 1) variables necessary to extract raw records from the SDD and to create combinations; and 2) variables necessary to create virtual records summarizing the combinations.

1. Combo Codes Input File

The Combo Codes Input File includes variables necessary to extract raw records from the SDD and to create combinations. This set of input variables will take the prefix "Raw". Those with unique values within a combo group will be in the Combo Input File whereas those defining a unique code value will be in the Combo Codes Input File.

Table 2: Combo Codes Input File Parameter Specification

Parameter	Field Name	Description
Raw Record Group	RawGroup	**Details**: Variable used to group different components that will define a combination item. **Note:** Similar to the GROUP variable in modular programs. This variable is used as the primary key to link the Combo Input File with the Codes Input File. **Defined by:** Request programmer **Input type:** Required **Format:** Character $30; no special characters (*e.g.,* commas, periods, hyphens, etc.) allowed, and underscores must be used to mark spaces. **Example**: RawGroup=AMI
Raw Record StockGroup	RawStockGroup	**Details**: This is the variable used to identify stockgroups of different components within a RawGroup. **Note:** Similar to the STOCKGROUP variable in modular programs, stockpiling will be executed at

Parameter	Field Name	Description
		this level. **Defined by:** Request programmer **Input type:** Required **Format:** SAS character $30; no special characters (*e.g.,* commas, periods, hyphens, etc.) allowed, and underscores must be used to mark spaces. **Example**: RawStockgroup=AMI1
Raw Extraction Code	RawCode	**Details**: This is the code used to extract the raw record from the SDD. The code will be defined in accordance with RawCodeType (below). • If RAWCODETYPE begins with DX then this field will include a diagnosis code. • If RAWCODETYPE begins with PX then this field will include a procedure code. • If RAWCODETYPE begins with RX then this field will include a NDC. • If RAWCODETYPE=EN01 then this field will include a length of stay in days. • If RAWCODETYPE=EN02 then this field will include a DRG. • If RAWCODETYPE begins with EL then this field is a number of continuous eligibility days. • If RAWCODETYPE begins with LAB01 then this field includes a lab code as specified in the Lab Code Map File. • If RAWCODETYPE begins with LAB02 then this field includes a LOINC. • If RAWCODETYPE begins with LABC4 then this field includes a CPT4 procedure code. • If RAWCODETYPE=DM01 then this field will include a sex value • If RAWCODETYPE=DM02 then this field will include a race value • If RAWCODETYPE=DM03 then this field will include a Hispanic value **Note 1:** For diagnosis codes, a wildcard (*) can be used to represent “any” value. For example, if RawCode = “410*” is specified as a diagnosis code, all codes that begin with “410” will be extracted. Wildcards are not supported for other RawCodeType values.

Parameter	Field Name	Description
		Note 2: Multiple wildcards cannot be specified in a code value (*e.g.,* "40*1*"). **Note 3:** Except for the "LOINC" laboratory extraction (RawCodeType= "LAB02"), ranges can also be specified using a hyphen. For example, if RawCode = "410.1-411.5" is specified, all codes between 410.1 and 411.5 (inclusive) will be extracted. Wildcards can also be used with ranges (*e.g.,* 401*-405*) for the lower bound and the upper bound for diagnoses. **Note 4:** When ranges are specified, wildcards are not allowed in the middle of the codes (e.g. 41*1-41*7 is not supported). However, wildcards are allowed if they are specified as the last character. **Note 5:** When ranges are specified, wildcards (if any) should be specified in the same position for the lower and upper bounds (*e.g.,* "410*- 412*" is correct whereas "410*-50*" is incorrect). **Note 6:** When sex values are specified, each should be enclosed by a single quotation mark and separated by a space. Example: 'F' 'M' **Defined by:** Request programmer **Input type:** Required **Format:** SAS character $30; no special characters (*e.g.,* commas, periods, hyphens, etc.) allowed, and underscores must be used to mark spaces. **Example**: RawCode=401.01
Raw Extraction Code Type	RawCodeType	**Details**: Indicates what type of record to extract and which type of code was supplied in RawCode. Allowable values are: • **DX09:** ICD9-CM diagnosis code • **DX10:** ICD10-CM diagnosis code • **DX11:** ICD11-CM diagnosis code • **PX09**: ICD-9-CM procedure code • **PX 10**: ICD-10-CM procedure code • **PX 11**: ICD-11-CM procedure code • **PX C4**: CPT-4 (*i.e.,* HCPCS Level I) procedure code • **PX HC**: HCPCS (*i.e.,* HCPCS Level II) procedure code

Parameter	Field Name	Description
		• **PX H3**: HCPCS Level III procedure code • **PX C2**: CPT Category II procedure code • **PX C3**: CPT Category III procedure code • **PX RE**: Revenue code • **RX09:** 9-digit NDC • **RX11:** 11-digit NDC • **EN01:** When the user needs to extract encounter records based on encounter type (SCDM EncType). The user can use RAWCODE to further specify the minimum length of stay (LOS) to be applied when extracting encounter records. The user can use RAWENCTYPE (below) to specify requested encounter types. • **EN02:** When the user needs to extract encounter records based on DRGs. DRG codes must be supplied in the RAWCODE field. • **EL01**: If the user needs to extract eligibility records based on MedCov=Y only. The minimum enrollment length can be supplied using the RAWCODE field. Do not use this RawCodeType with modular programs (modular programs already employ eligibility requirements as part of basic functionality). • **EL02:** If the user wants to extract eligibility records based on DrugCov=Y only. The minimum enrollment length can be supplied using the RAWCODE field. Do not use this RawCodeType with modular programs (modular programs already employ eligibility requirements as part of basic functionality). • **EL03:** If the user wants to extract eligibility records based on DrugCov=Y and MedCov=Y. The minimum enrollment length can be supplied using the RAWCODE field. Do not use this RawCodeType with modular programs (modular programs already employ eligibility requirements as part of basic functionality). • **LAB01N:** If the user needs to extract laboratory results based on the SOC Lab Code File and query quantitative results. • **LAB01C:** If the user needs to extract laboratory results based on the Lab Code File and query qualitative results.

Parameter	Field Name	Description
		• **LAB02N:** If the user needs to extract laboratory results based on LOINCs and query quantitative results. • **LAB02C:** If the user needs to extract laboratory results based on LOINCs and query qualitative results. • **LABC4N:** If the user needs to extract laboratory results based on procedure codes and query quantitative results. • **LABC4C:** If the user needs to extract laboratory results based on procedures codes and query qualitative results. • **DM01:** If the user needs to extract only specific sex demographic criteria. • **DM02:** If the user needs to extract only specific race demographic criteria. • **DM03:** If the user needs to extract only specific Hispanic demographic criteria. **Note 1:** For eligibility, records will be reconciled prior to creating combos. **Defined by:** Request programmer **Input type:** Required **Format:** Character $6 **Example1**: RawCodeType=EL02 **Example2**: RawCodeType=LABC4C
Raw Record Care Setting	RawCaresetting	**Details**: The encounter type(s) extracted for the combination item raw records. **Note1:** For laboratory result records only, this criterion will be applied to the SDD Pt_Loc variable with the following possible values: • **E:** Emergency department • **H:** Home • **I:** Inpatient • **O:** Outpatient • **U:** Unknown/Missing For diagnosis and procedure records, this criterion will be applied to the SDD EncType variable with the following possible values: • **IP:** inpatient hospital stays • **IS:** non-acute institutional stays

Parameter	Field Name	Description
		• **ED:** emergency department visits • **AV:** ambulatory visits • **OA:** other ambulatory visits **Note 2:** The envelope algorithm, if executed for diagnosis or procedure codes, is executed prior to record selection. **Note 3:** Each encounter type value must be specified between single quotes. **Defined by:** Request programmer **Input type:** Required (default value is blank to select any EncType) **Format:** Alphanumeric; A list can be provided using spaces between multiple EncTypes (commas are not allowed). **Example**: RawCaresetting= 'IP' 'IS' 'ED'
Raw Record Principal Diagnosis	RawPDX	**Details**: The PDX values extracted for the combination item raw records. Valid values include: • **P:** Principal • **S:** Secondary • **X:** Unable to Classify If left empty, all PDX values will be extracted. **Note 1:** Should only be populated with RawEncType= IP. If a value is specified when RawEncType is not IP, the RawPDX value will be ignored. **Defined by:** Request programmer **Input type:** Required (default value is blank to select any PDX) **Format:** Character $1 **Example**: RawPDX=P
Raw Record Laboratory Date Type	RawLabDateType	**Details**: Specifies in what sequence date(s) in the Laboratory Result table should be considered to select one relevant date for a laboratory result of interest. The parameter will allow the user to either specify 1) a single date variable (Lab_dt, Order_dt, or Result_dt) to use; or 2) a hierarchy to choose a date variable (*e.g.,* select Lab_dt else if missing select Result_dt else if missing select Order_dt). Valid values are any combination of the following:

Parameter	Field Name	Description
		• **L**: Lab Date • **O:** Order Date • **R:** Result Date **Defined by:** Request programmer **Input type:** Required for laboratories **Format:** Alphanumeric **Example**: RawLabDateType =LRO. In this case, the program will select Lab_dt else if missing select Result_dt else if missing select Order_dt.
Raw Record Laboratory Result	RawLabResult	**Details**: Specify a lab result value or lab result range for querying. The raw lab result field allows for values or ranges of quantitative laboratory results (*e.g.,* 100; 100-200) and values of qualitative laboratory results (*e.g.,* "POSITIVE"). Valid values are: • **<=X** (less than or equal to X) • **<X** (less than X) • **>=X** (greater than or equal to X) • **>X** (greater than X) • **~=X** (not equal to X) • **X:Y** (between X and Y) Any string of relevant characters is allowed for *qualitative* results querying. **Note 1**: There are two fields in the Laboratory Result table that include results: MS_Result_C (contains results for qualitative tests) and MS_Result_N (contains results for quantitative tests). The field where the result will be queried will depend on the RAWCODETYPE value. **Note 2**: Ranges cannot be specified with hyphens. Must use ":". **Defined by:** Request programmer **Input type:** Required for laboratory results **Format:** Alphanumeric **Example1**: RawLabResult=20:50 **Example1**: RawLabResult=Positive

2. Combo Input File

The Combo Input File includes variables necessary to create virtual records summarizing the combinations that will be ultimately used by programs as if they were typical records out of the SDD (*e.g.,* diagnosis, procedure or dispensing records). This set of input variables will take the prefix "Comb".

Table 3: Combo Input File Parameter Specification

Parameter	Field Name	Description
Combination Group	CombGroup	**Details**: This variable names the combo virtual record that is created. This field is not typically used by any modular program, but it is necessary to combine several combos in the same output file and to further identify them. **Note 1:** Defines unique records in conjunction with RawOrder. **Defined by:** Request programmer **Input type:** Required **Format:** Alphanumeric **Example**: CombGroup =SevAMI
Combination Processing Order	CombOrder	**Details**: The order in which combination items are processed. If a request programmer wants to use a combination item to define a subsequent combination item, the order must be specified so that a combination is created before it is used to define a subsequent combo. **Note 1:** Must start with the value 1. **Note 2:** This parameter is optional. If it is not included in the input file, the program will still execute without error. **Defined by:** Request programmer **Input type:** Optional **Format:** Numeric **Example**: 1
Combination Description	CombDescr	**Details**: Optional free text providing details for the CombGroup. **Defined by:** Request programmer **Input type:** Optional **Format:** Alphanumeric **Example**: CombDescr= AMI
Combination Code	CombCode	**Details**: User-defined "code" that will appear on the final virtual record. This is how virtual records are identified using programming tools (*e.g.,* modular programs). This code can be seen as similar to an

Parameter	Field Name	Description
		ICD-9/CPT4/HCPCS code to identify diagnosis/procedure records or an NDC to identify a drug record. **Defined by:** Request programmer **Input type:** Required **Format:** SAS character $6-$11, depending on how you wish the combo virtual record to behave (diagnosis and procedure=$6., Drugs=$11.); no special characters allowed (*e.g.,* commas, periods, hyphens, etc.), and underscores must be used to mark spaces. **Example**: CombCode=AMI1
Combination Code Behavior	CombBehavior	**Details**: This variable determines how the final virtual record will ultimately behave. Based on this variable, the final combo virtual record will have the same SDD variables as defined in the chosen SDD component. Three choices are possible: • DX: the virtual record should behave like a diagnosis record • PX: the virtual record should behave like a procedure record • RX: the virtual record should behave like an outpatient dispensing record **Note 1:** If the user creates virtual records that span more than one day (ADate < DDate) and wants them to behave like DX or PX, then one record per day will be output to the output file. **Note 2:** DX_CodeType or PX_CodeType will always take the value "CB" on the final virtual combo record. **Note 3:** A meaningful value of the SDD Dispensing table RxAmt variable cannot be determined when complex combinations are required to behave like dispensing records. Therefore, when a final virtual record behaves like a dispensing record, RxAmt is assigned a value of "1" and should not be interpreted as a meaningful metric in output. **Defined by:** Request programmer **Input type:** Required **Format:** Alphanumeric

Parameter	Field Name	Description
		Example: CombBehavior=DX
Combination Encounter Type	CombEncType	**Details**: EncType assigned to the final virtual record. If CombBehavior is DX or PX, then this variable can take one of the following values: • **IP:** Inpatient Hospital Stay • **IS:** Non-Acute Institutional Stay • **ED:** Emergency Department • **AV:** Ambulatory Visit • **OA:** Other Ambulatory Visit **Note 1:** This value is not processed by the Combo Tool but is only included on the final virtual record. It is the programming tool (*e.g.,* modular program) utilizing the Combo Tool that will ultimately process the final virtual record(s). Just like for SDD table observations, blank values are not allowed. If this variable is left empty, it is the program processing the final virtual records that will decide what to do. **Defined by:** Request programmer **Input type:** Required (but can be left blank) **Format:** Alphanumeric **Example**: CombEncType=AV
Combination Principal Diagnosis	CombPDX	**Details**: PDX value to be included on the final virtual record. If CombEncType=IP, then this variable can take one of the following values: • **P:** Principal • **S:** Secondary • **X:** Unable to Classify **Note 1:** Only relevant for IP records; otherwise, leave empty. Just like for the CombEncType variable, this value is not processed by the Combo Tool but is only included on the final virtual record. It is the programming tool (*e.g.,* modular program) utilizing the Combo Tool that will ultimately process the final virtual record(s). For example, if this variable is left empty when CombEncType=IP, the claim will be processed just like the SDD Diagnosis Table records for which EncType is IP and PDX is missing. **Defined by:** Request programmer **Input type:** Required (but can be left blank) **Format:** Alphanumeric

Parameter	Field Name	Description
		Example: CombPDX=P
Raw Record Group	RawGroup	**Details**: Variable used to group different components that will define a combination item. **Note 1:** Similar to the GROUP variable in modular programs. This variable is used as the primary key to link the Combo Input File with the Codes Input File. **Defined by:** Request programmer **Input type:** Required **Format:** Alphanumeric; SAS character $30; no special characters (*e.g.,* commas, periods, hyphens, etc.) allowed, and underscores must be used to mark spaces. **Example**: RawGroup=AMI
Raw Record Order	RawOrder	**Details**: Instructs the algorithm which order to apply the combination items in the two-by-two method. If there are two types of items to combine together (*e.g.,* drug + diagnosis), the drug can be processed prior to or after the diagnosis depending on which has a RawOrder value of 1. **Note 1**: The combination item processed first by the two-by-two method must have a RawOrder value of 1 and others values must follow sequentially and will be processed in the specified order. **Note 2:** Defines unique records in conjunction with CombGroup. **Defined by:** Request programmer **Input type:** Required **Format:** Numeric >=1; **Example**: RawOrder=2
Raw Episode Creation Type	RawEpiType	**Details:** Determines whether episodes of the combination item are created. An episode will consist of a single continuous interval of the combination item (with an interruption of no more than RawEpiGap days [below]). If episodes are created, the value of RawEpiType indicates how episodes will be created. *Valid values include*: • **0:** No episode of this combination item will be created • **1:** Episodes of this combination item will be created by *Patient* and *RawGroup*

Parameter	Field Name	Description
		• **2:** Episodes will be created by *Patient*, *RawGroup* and *RawStockgroup* **Defined by:** Request programmer **Input type:** Required **Format:** Numeric; **Example**: RawEpiType=1
Raw Episode Gap	RawEpiGap	**Details:** Sets the number of days allowed between two consecutive events to consider them as part of the same treatment episode. For a given event, a gap of more than RawEpiGap days between the event date and the date of last day of supply (or ADate or DDate) of the previous event, triggers a new treatment episode. For example, if RawEpiGap =10 and we are creating episodes of drug use with dispensing 1's last day of supply on 1/31/2012 and dispensing 2's start date on 2/12/2012, then the algorithm will end the current episode on 1/31/2012 and start a new treatment episode on 2/12/2012 because there are more than 10 days between the two dispensings. **Named by:** Requester **Input type:** Required (default value is 0) **Format:** Numeric **Example:** RawEpiGap=10
Raw Record Exclusion	RawExclusion	**Details**: Indicates whether the combination item is an exclusion criterion. *Valid values include*: • **0:** the combination item is not an exclusion criterion (default) • **1:** the combination item is an exclusion criterion **Note 1:** Cannot be used with RawOrder=1. **Defined by:** Request programmer **Input type:** Required (default value is 0) **Format:** Numeric **Example**: RawExclusion=1
Raw Record Window From Start Date	RawDaysFromStartDt	**Details**: Determines the interval of days in which a combination item (Raw Order=N) must overlap the previous combination item (RawOrder=N-1) **ADate** to be deemed a combination.

Parameter	Field Name	Description
		For example, RawDaysFromStartDt= -180, 0 for combination item RawOrder=N will request this combination item to overlap the 180-day period prior to the ADate of the previous combination RawOrder=N-1 item and up to the day of the ADate of combination itemRawOrder= N-1 (the 0 in the interval). **Note 1:** If left empty, then pure overlap (no extensions) will be deemed a valid combination. This is the same as specifying RawDaysFromStartDt=0, 0. **Note 2:** Can be open-ended. For example, if an overlap period can start 60 prior to the ADate of the previous item and go up to the EDate of the same item, the requester can specify RawDaysFromStartDt=-60. **Defined by:** Request programmer **Input type:** Required (can be left blank see note above) **Format:** Alphanumeric; interval separated by a comma **Example**: RawDaysFromStartDt=-180,30
Raw Record Window From End Date	RawDaysFromEndDt	**Details**: Determines the interval of days in which a combination item (Raw Order=N) must overlap the previous combination item (RawOrder=N-1) **EDate** to be deemed a combination. For example, RawDaysFromEndDt= 0,30 will request this combination item to overlap the 30 day period on or after the EDate of the previous combination. **Note 1:** If left empty, then pure overlap (no extensions) will be deemed a valid combination. This is the same as specifying RawDaysFromStartDt=0, 0. **Note 2:** Can be open-ended. For example if an overlap period can start the day of the ADate of the previous item and go up to 60 after the EDate of the same item, the requester can specify RawDaysFromEndDt=60. **Defined by:** Request programmer **Input type:** Required (can be left blank see note above) **Format:** Alphanumeric; interval separated by a

Parameter	Field Name	Description
		comma **Example**: RawDaysFromEndDt =-30,30
Raw Record Duration	RawDurat	**Details**: This parameter allows the user to add a fixed duration to a combination item raw record. This parameter is used in a way similar to the episode extension parameter used in some modular programs. **Defined by:** Request programmer **Input type:** Optional **Format:** Numeric (>=1) **Example**: RawDurat=30
Raw Record Start Date Type	RawADateType	**Details**: Defines the ADate to keep on the (interim or final) virtual record after having identified a valid combination of two (or more) items. Valid values include: • **ADATEB:** The start date of the Base interval • **ADATEC:** The start date of the Combined-to interval • **MINADATEBC:** The minimum date between the Base and Combined-to start date • **MAXADATEBC:** The maximum date between the Base and Combined-to start date • **EDATEB:** The end date of the Base interval • **EDATEC:** The end date of the Combined-to interval • **MINEDATEBC:** The minimum date between the Base and Combined-to end date • **MAXEDATEBC:** The maximum date between the Base and Combined-to end date • **MINOVER:** The start of the overlapping period between Base and Combined-to interval • **MAXOVER:** The end of the overlapping period between Base and Combined-to interval **Defined by:** Request programmer **Input type:** Required when RawOrder > 1 **Format:** Alphanumeric; SAS character $10 **Example**: RawADateType= MINADATEBC
Raw Record End Date Type	RawEDateType	**Details**: Defines the EDate to keep on the (interim) virtual record after having identified a valid combination of two (or more) items. Valid values include: • **ADATEB:** The start date of the Base interval

Parameter	Field Name	Description
		• **ADATEC:** The start date of the Combined-to interval • **MINADATEBC:** The minimum date between the Base and Combined-to start date • **MAXADATEBC:** The maximum date between the Base and Combined-to start date • **EDATEB:** The end date of the Base interval • **EDATEC:** The end date of the Combined-to interval • **MINEDATEBC:** The minimum date between the Base and Combined-to end date • **MAXEDATEBC:** The maximum date between the Base and Combined-to end date • **MINOVER:** The start of the overlapping period between Base and Combined-to interval • **MAXOVER:** The end of the overlapping period between Base and Combined-to interval **Note 1**: At the end of the two-by-two algorithm, if EDate>ADate and CombBehavior is not RX, then one record per day will be outputted. **Defined by:** Request programmer **Input type:** Required when RawOrder > 1 **Format:** Alphanumeric; SAS character $10 **Example**: RawEDateType= MINEDATEBC

C. STOCKPILING INPUT FILE

The Stockpiling Input File is used to instruct the Combo Tool on how valid dispensings are selected and used by the stockpiling algorithm to create treatment episodes. Requesters can require restrictions on days supplied and amount supplied values for dispensings that are considered by the Combo Tool, and determine how the program adjusts dispensing dates based on the amount of overlap between adjacent dispensings.

Table 4: Stockpiling Input File Parameter Specification

Parameter	Variable Name	Description
Name of Raw Group	RAWGROUP	**Details:** Standardized name used to refer to a RAWGROUP in the Combo Codes Input File. **Note 1**: Must match RAWGROUP values from the Combo Codes Input File. **Named by:** Request programmer **Input type:** Required **Format:** Alphanumeric; SAS character $30; no special characters (*e.g.,* commas, periods, hyphens, etc.) allowed, and underscores must be used to mark spaces. **Example:** RawGroup1
Same Day Dispensing Processing Indicator	SAMEDAY	**Details**: Defines how same day dispensings are processed. The first position indicates how days supplied (RxSup in the SCDM) is handled; the second position indicates how amount supplied (RxAmt in the SCDM) is handled. Valid values (for each position are): • **a:** adds all (amount supplied or days supplied) values for dispensings in the same GROUP/STOCKGROUP on the same day • **n:** uses minimum (amount supplied or days supplied) value for dispensings in the same GROUP/STOCKGROUP on the same day • **x:** uses maximum (amount supplied or days supplied) value for dispensings in the same GROUP/STOCKGROUP on the same day • **m:** uses mean (amount supplied or days supplied) value for dispensings in the same GROUP/STOCKGROUP on the same day **Note 1**: A total of 16 combinations are possible (*e.g.,* aa, an, etc.). **Note 2**: Default value is “aa”. **Defined by:** Requester **Input type:** Required

Parameter	Variable Name	Description
		Format: SAS character $2 **Example:** SAMEDAY = aa
Range of Allowable Days Supplied Values	SUPRANGE	**Details:** Specifies the allowable range of days supplied values (variable RxSup in the SCDM) that are allowed for a dispensing to be used to create valid treatment episodes. Valid values are: • **x<-HIGH:** value must be > x • **y-HIGH**: value must be >= y • **LOW-<x**: value must be < x • **x-y**: value must be between x and y inclusively • **x<-y**: value must be greater than x and less or equal than y • **x-<y**: value must be greater or equal than x and less than y • **x<-<y**: value must be between x and y but not equal **Note 1**: Allowable values can also be discrete, *e.g.,* "10", "20". **Note 2**: Failing to be in the specified range excludes a dispensing from consideration. **Note 3:** Default is "0<-HIGH", indicating that the program will not consider days supplied values of 0 or less. **Defined by**: Requester **Input type**: Required **Format**: SAS character $40 **Examples**: SUPRANGE=5-<80; SUPRANGE = 0<-HIGH
Range of Allowable Amount Supplied Values	AMTRANGE	**Details:** Specifies the allowable range of amount supplied values (variable RxAmt in the SCDM) that are allowed for a dispensing to be used to create valid treatment episodes. Valid values are: • **x<-HIGH:** value must be > x • **y-HIGH**: value must be >= y • **LOW-<x**: value must be < x • **x-y**: value must be between x and y inclusively • **x<-y**: value must be greater than x and less or equal than y • **x-<y**: value must be greater or equal than x and less than y • **x<-<y**: value must be between x and y but not equal **Note 1**: Allowable values can also be discrete, *e.g.,* "10", "20". **Note 2**: Failing to be in the specified range excludes a dispensing from consideration. **Note 3:** Default is "0<-HIGH", indicating that the program will not

Parameter	Variable Name	Description
		consider amount supplied values of 0 or less. **Defined by**: Requester **Input type**: Required **Format**: SAS character $40 **Examples**: SUPRANGE=5-<80; SUPRANGE = 0<-HIGH
Overlap Percentage Processing	PERCENTDAYS	**Details:** The maximum percentage overlap of previous dispensing's days supply allowed for pushing dispensing dates forward. When this percentage is exceeded, the previous dispensing's days supply is truncated at the day prior to the next dispensing date. If this parameter is left blank, no truncation will occur and any overlap of supply between dispensing will be corrected by pushing overlapping days supplied forward. **Note 1**: Default is blank. **Note 2**: Although rare, when PERCENTDAYS >0 it is possible for the overlap to exceed 100%. If this occurs, the dispensing will be replaced by the eclipsed claim. Defined by: Requester Input type: Optional Format: Numeric Example: PERCENTDAYS = 0.25

D. LAB CODE MAP INPUT FILE

The Lab Code Map File is a master lookup file of valid lab codes developed and maintained by the SOC and the Clinical Data Workgroup. As the workgroup continues to develop and enhance the Laboratory Result table specifications and allowable values, this master lookup file can be modified to ensure that the Combo Tool always queries the SCDM Laboratory Result table based on the current specifications.

This file includes a comprehensive list of all SOC defined "lab codes" denoting allowable combinations of lab test name, lab test subcategory, specimen source, result type, fasting indicator, patient location, and result unit. Lab codes are 14-digit identifiers developed by the SOC to represent a unique laboratory test result value for querying. The first digit of this code is an "L" indicative of a lab code, digits 2-4 indicate a unique lab test name, digit 5 indicates a unique result type value (numeric or character), digits 6-7 indicate a unique lab test subcategory, digit 8 represents a unique fasting indicator value, digits 9-10 indicate a unique specimen source, digits 11-12 indicate a unique patient location, and digits 13-14 indicate the result unit.

As SOC continues to develop and expand the SCDM Laboratory Result table and allowable values, this master lookup file may be modified to ensure that the program always queries the SCDM Laboratory Result table based on current specifications.

The Lab Codes defined in this lookup table are used to query the lab test and results of interest just as any NDC, diagnosis, or procedure code is queried in the SDD.

Table 5: Lab Code Map Input File Parameter Specification

Parameter	Variable Name	Description
Lab Test Name	Ms_Test_Name	**Details:** Value of the lab test name parameter in the SCDM Laboratory table. **Defined by:** SOC **Input type:** Required **Format:** SAS character $10. **Example:** ALP
Lab Test Subcategory	Ms_Test_Sub_Category	**Details:** Value of the lab test subcategory parameter in the SCDM Laboratory table. **Defined by:** SOC **Input type:** Required **Format:** SAS character $6. **Example:** NS
Lab Specimen Source	Specimen_Source	**Details:** Value of the lab specimen source parameter in the SCDM Laboratory table. **Defined by:** SOC **Input type:** Optional **Format:** SAS character $6. **Example:** BAL
Lab Result Unit	Ms_Result_Unit	**Details:** Value of the lab result unit parameter in the SCDM Laboratory table. **Defined by:** SOC **Input type:** Required **Format:** SAS character $11. **Example:** PERCENT
Result Type	RESULT_TYPE	**Details:** Value of the result type in the SCDM Laboratory Results table. **Defined by:** SOC **Input type:** Required **Format:** SAS character $1. **Example:** N
Fasting Indicator	FAST_IND	**Details:** Value of the fast_ind parameter in the SCDM Laboratory table. **Defined by:** SOC **Input type:** Required **Format:** SAS character $1.

Parameter	Variable Name	Description
		Example: R
Patient Location	PT_LOC	**Details:** Value of the pt_loc parameter in the SCDM laboratory table **Defined by:** SOC **Input type:** Required **Format:** SAS character $1. **Example:** R
Lab Code	Code	**Details:** SOC-defined code indicative of the MS_Test_Name, Result_Type, MS_Test_Sub_Category, Fast_Ind, Specimen_Source, Pt_Loc and MS_Result_Unit combination. **CODE values in the lookup table should contain an exhaustive list of all combinations of these variable values.** **Note 1**: CODE values can be listed in program Input Files to query the desired laboratory result values, just as any other NDC, diagnosis and/or procedure code is queried. **Defined by:** SOC **Input type:** Required **Format:** SAS character $14. **Example:** L0092012010205

IV. OUTPUT

If the SAVETODPLOCAL macro parameter is set to Y, up to three output files can be created in the DPLocal library using the OUTNAME macro parameter value.

A. [OUTNAME]_PROC

The [OUTNAME] _PROC dataset is created when final virtual records are required to behave like procedure records. This dataset mimics the structure of the SDD Procedure table:

CombGroup	CombDescr	PatID	ADate	EncType	PX	PX_CodeType	PDX
PROC1		*PatID1*	*01/01/13*	*IP*	*[COMBCODE]*	*CB*	*P*
PROC1		*PatID2*	*04/13/13*	*IP*	*[COMBCODE]*	*CB*	*P*

This allows other programming tools to query the combination items just as they would the SDD Procedure table.

B. [OUTNAME]_DIAG

The [OUTNAME] _DIAG dataset is created when final virtual records are required to behave like diagnosis records. This dataset mimics the structure of the SDD Diagnosis table:

CombGroup	CombDescr	PatID	ADate	EncType	DX	DX_CodeType
DIAG1		*PatID3*	*02/04/12*	*AV*	*[COMBCODE]*	*CB*
DIAG1		*PatID4*	*07/17/12*	*AV*	*[COMBCODE]*	*CB*

This allows other programming tools to query the combination items just as they would the SDD Diagnosis table.

C. [OUTNAME]_DRUG

The [OUTNAME] _DRUG dataset is created when final virtual records are required to behave like dispensing records. This dataset mimics the structure of the SDD Dispensing table:

CombGroup	CombDescr	PatID	RxDate	NDC	RxSup	RXAmt
DRUG1		*PatID5*	*03/09/13*	*[COMBCODE]*	*120*	*1*
DRUG1		*PatID6*	*08/27/12*	*[COMBCODE]*	*30*	*1*

This allows other programming tools to query the combination items just as they would the SDD Dispensing table.

D. WORK DIRECTORY FILES

To allow a simple integration of this tool in the modular programs and to avoid querying the SDD tables more than once, three datasets are created in the work directory. Even if these datasets are not used by the Combo Tool, they are created intentionally:

- _ProcExtract
- _DiagExtract
- _Drugs