

Sentinel Quality Assurance Lookup Table Generator Tool Functional Specification

0

Prepared by the Sentinel Operations Center

April 03, 2024

Sentinel Functional Specification for Local QA Lookup Table Generator Tool

Table of Contents

[1. Introduction 1](#_Toc141429689)

[1.1. Purpose 1](#_Toc141429690)

[1.2. Terminology 1](#_Toc141429691)

[1.3. Summary 1](#_Toc141429692)

[1.4. Requirements 2](#_Toc141429693)

[2. System Description 2](#_Toc141429694)

[2.1. Operating System 2](#_Toc141429695)

[2.2. Software 2](#_Toc141429696)

[3. Package Requirements 2](#_Toc141429697)

[3.1. Location 2](#_Toc141429698)

[3.2. Naming Convention 2](#_Toc141429699)

[3.3. Folder Structure 2](#_Toc141429700)

[4. Output 3](#_Toc141429701)

[4.1. MSOC 3](#_Toc141429702)

[4.2. DPLOCAL 3](#_Toc141429703)

[4.2.1. check\_key.sas7bdat 3](#_Toc141429704)

[4.2.2. master\_all\_flags.sas7bdat 4](#_Toc141429705)

[4.2.3. master\_all\_flags.xlsx 6](#_Toc141429706)

[4.2.4. master\_key.sas7bdat 6](#_Toc141429707)

[5. Input 7](#_Toc141429708)

[5.1. SASPROGRAMS 7](#_Toc141429709)

[5.1.1. soc\_qa\_lookup\_master.sas 7](#_Toc141429710)

[5.2. INPUTFILES 7](#_Toc141429711)

[5.2.1. 00.1\_soc\_macros.sas 7](#_Toc141429712)

[5.2.2. 01.1\_create\_lookup\_tables.sas 8](#_Toc141429713)

[5.2.3. SCDM lookup table values.xlsx 8](#_Toc141429714)

History of Modifications

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | Modification | Author |
| 8.9.0 | 04/02/2024 | * Update control flow to create dummy records for SCDM tables not assessed by package and include Feature Engineer table | Sentinel Operations Center |
| 8.8.1 | 12/20/2023 | * Introduced use of JSON file * Removed references to MODULE\_CAT and MODULE\_UTIL | Sentinel Operations Center |
| 8.7.1 | 10/25/2023 | * Updated lab result lookups with additional SARS\_COV\_2 LOINC | Sentinel Operations Center |
| 8.7.0 | 08/31/2023 | * Updated documentation * Consolidated QAR and QCR Lookups | Sentinel Operations Center |
| 8.6.3 | 07/20/2023 | * Removed duplicate checks | Sentinel Operations Center |
| 8.6.2 | 03/24/2023 | * Bug Fixes | Sentinel Operations Center |
| 8.6.1 | 03/24/2023 | * Updated LOINC list for mapped lab tests in lab table based on Regenstrief release | Sentinel Operations Center |
| 8.6.0 | 02/23/2023 | * Introduced new FlagType NOTE | Sentinel Operations Center |
| 8.5.0 | 09/22/2022 | * Addition of checks for decimals in ICD px and dx codes in PRO and DIA tables * Quarterly LOINC list update for mapped lab tests in LAB table * Minor bug fixes to address SCDM v8.1.0 upversioning | Sentinel Operations Center |
| 8.4.2 | 06/14/2022 | * Minor bug fix | Sentinel Operations Center |
| 8.4.1 | 06/07/2022 | * Minor bug fix | Sentinel Operations Center |
| 8.4.0 | 05/25/2022 | * Major enhancements for SCDM v 8.1.0 upversioning | Sentinel Operations Center |
| 8.3.5 | 03/22/2022 | * Update valid values in PRO and LAB table * Modifications to account for new CPT-4 code formats | Sentinel Operations Center |
| 8.0.0 | 9/23/2021 | * Add level 2 look-up table to identify invalid value combinations in encounter table * Modify flags to add new level 2 encounter table invalid combination of valid value checks | Sentinel Operations Center |
| 7.0.3 | 02/22/2021 | * Minor change to input file for lkp\_all\_l1 to update Inpatient Transfusion variable Orig\_TransProd length definition to 0 (definition of DP site specific length) * Update to IRX CheckID 206 from FAIL to WARN * Update CheckID 111 on optional TXN variables. | Sentinel Operations Center |
| 7.0.2 | 02/18/2021 | Minor changes to create “soft fails” (no abort) for model compliance issues related to SCDM v8.0.0 Prescribing table | Sentinel Operations Center |
| 7.0.1 | NA | Skipped, and combined all changes into 7.0.2 | Sentinel Operations Center |
| 7.0.0 | 02/09/2021 | Version update for SCDM v8.0.0 | Sentinel Operations Center |
| 6.2.0 | 03/10/2020 | Version update for qa\_package\_6.2.0 | Sentinel Operations Center |
| 6.1.0 | 11/27/2019 | Original Version | Sentinel Operations Center |

Sentinel Standard Abbreviations

|  |  |
| --- | --- |
| Acronym | Description |
| DMQA | Data Management & Quality Assurance |
| DP | Data Partner Identifier |
| ETL | Extract, Transform, Load |
| Phase | ETL phase (A or B) |
| QA | Quality Assurance |
| SAS® | Statistical Analysis Software |
| SCDM | Sentinel Common Data Model |
| SOC | Sentinel Operations Center |

# Introduction

## Purpose

This document describes the internal program package (hereafter referred to as "package") used by the Sentinel Operations Center (SOC) Data Management and Quality Assurance (DMQA) team for creation of "lookup" SAS datasets to be used programmatically by the distributed Quality Assurance (QA) Phase A program package and for internal reference by SOC. These datasets contain table, variable, and value requirements based on the current Sentinel Common Data Model (SCDM), as well as the master list of QA data checks.

## Terminology

* Data Partner (DP): An organization that contributes data to the Sentinel Program
* Sentinel Common Data Model (SCDM): The structure of the data that all Sentinel DPs are required to adhere to participate in the Sentinel Program
* Package: The final executable product, in the Sentinel standard folder structure and containing all necessary program files and data files.
* Distributed: Used to describe a package that is created by the SOC and executed by DPs at their site
* Local: Used to describe a package that is created by the SOC for internal use only.
* Extract, Transform and Load (ETL): Refers to a process in database usage and especially in data warehousing that:
  + [Extracts](https://en.wikipedia.org/wiki/Data_extraction) data from homogeneous or heterogeneous data sources;
  + [Transforms](https://en.wikipedia.org/wiki/Data_transformation) the data for storing it in the proper format or structure for the purposes of querying and analysis; and
  + [Loads](https://en.wikipedia.org/wiki/Data_loading) it into the final target (database, more specifically, [operational data store](https://en.wikipedia.org/wiki/Operational_data_store), [data mart](https://en.wikipedia.org/wiki/Data_mart), or [data warehouse](https://en.wikipedia.org/wiki/Data_warehouse))
* QA package: The distributed package that generates a high volume of datasets that are used for the Quality Assurance process. These datasets contain statistical information/counts of how every variable is populated at a DP.
* Phase A / Phase B: Since the introduction of the Mother-Infant Linkage (MIL) table in SCDM v7.0.0, ETL Refreshes have been completed in phases by some DPs. Phase A QA Package evaluates the core and optional SCDM tables, while the Phase B QA Package evaluates the MIL table constructed after Phase A has been approved. While similar in process, this document is specific to Phase A lookup table creation.

## Summary

This local SAS package, executed at SOC, will create the tables needed by the distributed Phase A QA package to validate a Data Partner's database against the current SCDM for model compliance. It is executed locally on an as-needed basis.

In addition, this package generates documentation detailing all Level 1, Level 2, and Level 3 flags used by the QA package and Sentinel Analysts for assessing the quality of a Data Partner's data refresh.

## Requirements

This local SAS package should create the tables that will be used programmatically as input files in the QA package. The primary user of the package is a SOC programmer and/or analyst who needs to review the current SCDM and update the qa\_lookup input file, as necessary. This may include updating or adding a table, variable, or valid value based on a model change; or updating, adding, or removing a data check flag.The SOC user should have the option to specify parameters of interest within the package.

# System Description

## Operating System

The required operating system is Microsoft Windows.

## Software

This package will be created using SAS and should be run in batch mode with SAS version 9.4 or higher. Microsoft Excel for Office 365 32-bit (at minimum) is also required.

# Package Requirements

## Location

The final package will be stored in Git version control as “qa\_lookup” and development and QC will be tracked in JIRA, using the component “qa\_lookup”.

## Naming Convention

The package will be referenced as “qa\_lookup\_{version} (e.g., qa\_lookup\_1.0.0).

## Folder Structure

The standard SOC folder structure should be used for creating this package, with the following subfolders:

* dplocal:

The ‘dplocal’ subfolder is for output generated by the package that should remain with the Data Partner (and may be used to facilitate follow-up queries). For internal packages, this folder may be used for temporary and/or permanent output.

* inputfiles:

The read-only ‘inputfiles’ subfolder contains additional program files and lookup tables needed to execute the package.

* msoc:

The ‘msoc’ subfolder is for output generated by the package that will remain with SOC.

* sasprograms:

This read-only subfolder contains the master SAS program that must be edited and then executed by the SOC analyst.

# Output

## MSOC

The datasets output to the MSOC folder of this program are documented as [input files in the QA Package documentation](https://dev.sentinelsystem.org/pages/SENTINEL/sentinel-quality-assurance-documentation/master/browse/qar-inputfiles.html).

## DPLOCAL

There are three (3) SAS tables and one (1) Excel workbook that are output to the “dplocal” subfolder.

### check\_key.sas7bdat

This SAS output table contains the master list of QA data checks that are available for use.

| Variable Name | Format | Valid Values | Notes/Description | Example(s) |
| --- | --- | --- | --- | --- |
| Level | $1. | 1, 2, 3, 4\* | FlagID token indicating Level of data check | 1 |
| CheckID | $3. | {###} | FlagID token indicating unique data check identifier | 100 |
| Table1 | $3. | {XXX} | SCDM table abbreviation | COD |
| Table2 | $3. | 0, {XXX} | SCDM table abbreviation | 0 |
| \_VarNum1 | $46. | Free text | Unique variable identifier, including variable type/condition | 0 |
| VarNum2 | $2. | {##} | Unique variable identifier | 0 |
| VarNum3 | $2. | {##} | Unique variable identifier | 0 |
| VarNum4 | $2. | {##} | Unique variable identifier | 0 |
| \_TestNum | $48. | Free text | Laboratory test type/condition | 0 |
| \_Result\_type | $20. | Free text | Laboratory test result type/condition | 0 |
| AbortYN | $1. | N, Y | Indicator for implication of the data check for the QA package. an ETL under review. Successful execution of the QA package requires no AbortYN=Y data error(s). | Y |
| FlagType | $4. | Fail, Warn | Indicator for implication of the data check for passing or failing an ETL under review. A “Fail” flag means the ETL cannot pass QA review until fixed. A “Warn” flag is for characterization or reference purposes. | Fail |
| FlagYN | $1. | N, Y | Indicates whether the data check is automatically executed by QA package or not. Most Level 1 and Level 2 checks will have FlagYN=Y. | Y |
| Var1 | $21. | na, SCDM variable name as it appears in the model | Primary variable being checked | na |
| Var2 | $21. | NA, SCDM variable name as it appears in the model | Second variable being checked | NA |
| CheckID\_Description | $178. | Free text | Description of data check | &table1. table does not exist |
| DatasetIn | $37. | NA, dplocal.{QA package dataset name}, msoc.{QA package dataset name} | Indicates the QA package output dataset used programmatically as the base of the check. This may be deleted when no longer needed for processing. Enter “NA” if not applicable. | NA |
| DatasetOutQA | $33. | NA, dplocal.{QA package output dataset name}, msoc.{QA package output dataset name} | Indicates dataset output by QA package to ‘msoc’ or ‘dplocal’. Enter “NA” if not applicable. | dplocal.all\_l1\_flags |
| DatasetOutComp | $35. | NA, {compare package output dataset name} | Indicates dataset output from QA package that is used in local Compare Package for cross-ETL analysis Enter “NA” if not applicable. | NA |
| Lookup\_table | $18. | Free text | Indicates the name of the lookup table required for the check. Enter “NA” if not applicable. | NA |

\* Level 4 checks have been removed from the QA package temporarily but will be added back in the future.

### master\_all\_flags.sas7bdat

This SAS output table contains the all-inclusive master list of QA data checks.

| Variable Name | Format | Valid Values | Notes/Description | Example(s) |
| --- | --- | --- | --- | --- |
| FlagID | $21. | {TableID}\_{Level}\_{VarID}\_{TestID}\_{CheckID} | Error code/QA data check | ENC-ENR\_2\_01\_00-0\_204 |
| FlagType | $4. | Fail, Warn | Type of flag, whether it fails or not | Warn |
| AbortYN | $1. | Y(es), N(o) | Indicates whether QA module will abort or not | N |
| TableID | $7. | {XXX} or {XXX-XXX} | SCDM table abbreviation | ENC-ENR |
| Level | $1. | 1, 2, 3, 4\* | Level of data check | 2 |
| VarID | $2. | {##} | Unique variable identifier | 01 |
| TestID | $4. | {##}-{N(umeric) or C(haracter)} | Laboratory test identifier | 00-0 |
| CheckID | $3. | {###} | Unique data check identifier | 204 |
| Variable1 | $21. | SCDM variable name as it appears in the model | Primary variable being checked | PatID |
| Variable2 | $21. | SCDM variable name as it appears in the model | Second variable being checked | NA |
| Variable3 | $21. | SCDM variable name as it appears in the model | Third variable being checked | NA |
| Variable4 | $21. | SCDM variable name as it appears in the model | Fourth variable being checked | NA |
| MS\_Test\_Name | $10. | SCDM Lab table test name as it appears in the model | Laboratory test name | NA |
| Result\_type | $1. | 0, N(umeric) or C(haracter) | Laboratory test result type | 0 |
| Flag\_Descr | $255. | Free text | Description of data check | PatID value length distribution is not consistent between tables |
| Dataset | $50. | Free text | Intermediate, temporary dataset used by QA package that may be deleted when no longer needed for processing | dplocal.all\_l2\_crosstab\_length\_value |
| DatasetOutQA | $50. | NA, dplocal.{QA package output dataset name}, msoc.{QA package output dataset name} | Dataset that is output by QA package to ‘msoc’ or ‘dplocal’ | msoc.all\_l2\_crosstab\_length\_value |
| DatasetOutComp | $50. | NA, {compare package output dataset name} | Dataset output from QA package that is used in local Compare Package for cross-ETL analysis | l2\_length\_value\_patid |
| Lookup\_Table | $20 | NA, {lookup table name} | Lookup table to reference for data check | NA |
| FlagYN | $1. | N, Y | Indicates whether the data check is automatically validated by QA package or not | N |
| SortOrder | 3. | {#} | Used to sort FlagIDs by CheckID-Table-Variable-Value | 471 |

\* Level 4 checks have been removed from the QA package temporarily but will be added back in the future.

### master\_all\_flags.xlsx

This Excel file contains the all-inclusive master list of QA data checks; it contains the same, exact content as master\_all\_flags.sas7bdat, but is output into a different file type, an Excel workbook (.xlsx).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Variable Name | Format | Valid Values | Notes/Description | Example(s) |
| As described in DPLOCAL. master\_all\_flags.sas7bdat | As described in DPLOCAL. master\_all\_flags.sas7bdat | As described in DPLOCAL. master\_all\_flags.sas7bdat | As described in DPLOCAL. master\_all\_flags.sas7bdat | As described in DPLOCAL. master\_all\_flags.sas7bdat |

### master\_key.sas7bdat

This SAS output table contains one row per available QA data check ID. It includes checks that have already been pre-defined and checks that are available for future use. In addition, each row contains the level of the check ID, the check description, the check type (e.g., table, variable, value, etc.), and any supplementary notes (e.g., Laboratory Result table only checks, etc.).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Variable Name | Format | Valid Values | Notes/Description | Example(s) |
| Level | BEST12. | 1, 2, 3, . | Level of data check | 2 |
| CheckID | $3. | {###} | Unique data check identifier | 203 |
| Check\_Descr | $158. | Free text | Description of data check | &variable1. variable length is not consistent between tables |
| CheckType | $62. | Free text | Type of check | Table-to-table check |
| Notes | $28. | Free text | Notes | Site-specific variables only |

# Input

## SASPROGRAMS

### soc\_qa\_lookup\_master.sas

This is the package master program that requires user-specified input. After editing the required parameters, the user should execute this program in batch mode in order to run the package.

|  |  |  |
| --- | --- | --- |
| Parameter | Field Name | Details |
| QA Lookup Package root | root | **Description:** Path to the location of the QA Lookup SAS package  **Input type:** text  **Format:** char  **Example:** A:\dev\git\qa\_lookup\ |
| SAS zip code lookup file name | version | **Description:** Versioned name for the most current SAS supplied zip code lookup input file; file extension is not required  **Input type:** text  **Format:** char  **Example:** Jul19\_v9 |
| Quality Assurance (QA) Package Version ID | QAver | **Description:** Version identifier for the Quality Assurance (QA) Package using the output from this program  **Input type:** text  **Format:** char  **Example:** 7.0.0 |
| Sentinel Common Data Model (SCDM) Version ID | SCDMver | **Description:** Version identifier for the Sentinel Common Data Model (SCDM) version evaluated in the corresponding QA package.  **Input type:** text  **Format:** char  **Example:** 8.0.0 |

## INPUTFILES

### 00.1\_soc\_macros.sas

This program contains SAS utility macros that are used for automating functionality in the package. The following macro programs are defined in this file.

|  |  |  |
| --- | --- | --- |
| Macro Program | Parameter(s) | Details |
| Import\_xlsx | File | **Functionality:** Uses proc import to read specified xlsx file into sas7bdat format; resulting data set is placed in work library; label attribute is removed from the resulting data set.  **Call example:**  **%*import\_xlsx*** (file=%str(&infolder.\SCDM\_lookup\_table\_values)); |
| Test | Result\_type var1 var2 | **Functionality:** Creates unique Lab test name  **Call Example:** %*test* (result\_type=C, var1=MS\_Result\_C, var2=Modifier); |
| macrovar | var | **Functionality:** Creates program-specific global macro variables.  **Call Example:** %*macrovar* (var=modifier\_n); |

### 01.1\_create\_lookup\_tables.sas

This program is the package engine that produces each output Excel file and SAS data set, listed in the Output section below.

### SCDM lookup table values.json

This developer-specified JSON file is used to supply the input values needed by the program modules to create the final “lookup” SAS datasets used programmatically by the QA package. This file contains all of the lookup files in a JSON-containerized format, nineteen (19) containers in total, which feed the creation of the input files needed for the QA package to run. The file allows the developer to specify characteristics within each look up file (container)and is updated as necessary. For example, this may include updating or adding a container in the file based on a SCDM model change or the need to update, add, or remove a data quality check. Consider utilizing a [JSON editor](https://jsoneditoronline.org/) if updates need to be made.

#### lkp\_lab\_test

This container contains metadata that will create the *msoc.lk\_lab\_results* SAS dataset, as defined in section 4.1 . It contains a numeric index of all evaluated lab tests from the optional SCDM Laboratory Results table. If adding or removing new lab test results, the analyst should place the new entry at the bottom of the list using the next available sequence number (**TestNum**); enter **Result\_Type**as *C* if the result is expected to be a character string, or *N*if the expected result is numeric; The combination of **TestNum + Result\_Type** is considered the key identifier for the lab test when merging with the SCDM Labs table. Note, some tests can have both a numeric and a character result type and are therefore considered different “tests”. Once a test number is assigned, it is permanent; even if the test is removed from the SCDM, its number should never be re-used.

Enter the abbreviated test name in uppercase as **MS\_Test\_Name**(example, *CHOL\_LDL*). **UnitReq** determines whether the test unit is required in results, and should be coded as *Y* (yes), *N* (unit should be blank), or *U* (unit is not required but may exist). The final column **Characterized** identifies whether the test has been standardized (*I* or *N*). Note that if **Characterized** is *N*, **UnitReq**should be coded *U*.

#### lkp\_all\_l1

The lkp\_all\_l1 container contains metadata that will create the *msoc.lkp\_all\_l1* SAS dataset, as defined in section 4.1above. This worksheet contains one row per SCDM table and variable, listing the criteria for Level 1 compliance to the SCDM. To add, remove, or modify an entry, the following column definitions should be adhered to.

* **SCDMID** field contains an identifier for the SCDM Table in accordance to the SCDM Model Document (MD) located [here](https://wiki.sentinelsystem.org/display/DE/DE+Model+Documents)
* **TabID** field has the three-letter table identifier, in uppercase. Excel data validation function allows selection from the following list sourced from the \_list worksheet:
  + *COD*: cause of death
  + *DEM*: demographic
  + *DIA*: diagnosis
  + *DIS*: dispensing
  + *DTH*: death
  + *ENC*: encounter
  + *ENR*: enrollment
  + *LAB*: laboratory results
  + *PRO*: procedure
  + *VIT*: vital signs
  + *IRX*: inpatient pharmacy
  + *TXN*: inpatient transfusion
  + *PRE*: prescribing
  + *FAC*: facility
  + *PVD*: provider
* **VarID** column contains an arbitrarily assigned numeric identifier of the individual SCDM variable. The variable PatID should always be assigned as “01” and encounterID, if applicable, should always be assigned as “02”. No **VarID** is reused for retired or removed variable.
* **Variable** columncontains the name of the variable in the SCDM table specified
* **VarType** and **VarLength**fields indicate the expected variable type (*C* for character, *N* for numeric) and length (in bytes) of the variable listed above. Drop down options for valid **VarType** values are populated from \_List worksheet, column L (**ValidValueType**)
* **ObsReq**columndenotes the number of values for the variable that are expected to be populated. Valid values are derived from the \_List worksheet, column K (*ALL*, *SOME*, or *NONE*).

|  |
| --- |
| **Example 1**: **TabID** = *DEM* with **VarID** = *02* is a check on variable = *Birth\_Date*; This variable is numeric and has a length of 4. Since **ObsReq** = *SOME*, the Level 1 check defined by this row will evaluate for at least 1 observation populated for Birth\_Date variable. |

* **KeyVar**columnvalue of K indicates that the variable listed is a primary key or an attribute of a composite key; otherwise, blank.
* **CrossVar**field value of X indicates that this variable links across tables
* **ValidValueType** column defines the type of valid values for the given variable. The \_LIST worksheet, column L, contains the selection of valid value types available to select.
* **\_ValidValue\*\*** column value is derived from a formula based upon the selected **ValidValueType***,* as well as any values entered in columns K through R of this worksheet. The values of these columns will be populated in correlation to the **ValidValueType**listed.
* **\_date, \_time\*\*, \_numeric\*\*, \_numrange\*\*, \_case\*\*, \_find\*\*, \_only\*\*,** and **\_listvalues\*\*** columns are used to populate the **\_ValidValue** in accordance to the conditions defined in that column’s Excel function. These columns are relegated to drop down menu values available in the \_List worksheet.
* **\_TabVar\*\*** column value is automatically created as a concatenation of **TabID** + **VarID**. This column mirrors column S (**TabIDVarID**) in the \_List worksheet.

*\*\*these are sources for derived variables in the final output dataset*

|  |
| --- |
| **Example 2.1** - using **TabID** = *DIA* and **VarID** = *05* (variable = EncType): **ValidValueType**= *list\_values* and **\_listvalues** = *AV|ED|IP|IS|OA*. The type of check defined evaluates the variable EncType in the diagnosis table for the values (AV, ED, IP, IS, OA) and will flag any values not equal to those defined. |
| **Example 2.2** - using **TabID** = *DIS* and **VarID** = *04* (variable = RxSup): **ValidValueType**= *numeric* and **\_numeric** = *ge 0*. The type of check defined evaluates the variable RxSup in the dispensing table for a number greater than or equal to 0 and will flag any values not equal to those defined (such as negative numbers). |
| **Example 2.3** – using **TabID** = *VIT* and **VarID** = *11* (variable = Tobacco): **ValidValueType** = *num\_range* and **\_numrange** = *1:7*. The type of check defined evaluates the variable Tobacco from the vitals table for a number within the range of 1 to 7 and will flag any values outside of that defined range. |

#### lkp\_lab\_l2

The generated SAS lookup file that originates from this container is used in the level 2 evaluation of the SCDM Lab tables. There is one row per unique **MS\_Test\_Name**, **Result\_Type**, **MS\_Test\_Sub\_Category**, and **Fast\_Ind** value per the SCDM. When the package is executed, an individual observation is output for each delimited value in the **\_Specimen\_Source** field by row.

For assistance in assigning standard values in this workbook, please see [SCDM Lab Results Table Documentation](https://www.sentinelinitiative.org/sites/default/files/data/distributed-database/Sentinel_Common-Data-Model_Laboratory-Result-Table-Documentation_0.pdf).

* **MS\_Test\_Name** column contains a standardized test indicator in capitals.
* **Result\_Type** column indicates whether the expected test result value is character (*C*) or numeric (*N*).
* **\_MS\_Test\_Sub\_Category\*\*** field specifies any existing test subcategory
* **\_Specimen\_Source\*\*** column contains and pipe-delimited (‘|’) list of values for the specimen source from which lab results can derive. These values should be entered in all capitals and typically contain a value of *UNK* within the list.
* **Fast\_Ind** field contains an indicator of whether fasting was required at time of test.

*\*\*these are sources for derived variables in the final output dataset*

|  |
| --- |
| Example 1.1 – **MS\_Test\_Name=** *GLUCOSE*, **Result\_Type**=*N*, **\_MS\_Test\_Sub\_Category**= <*null>*, **\_Specimen\_Source**=*BLOOD|PLASMA|SERUM|SR\_PLS|UNK*, **Fast\_Ind**= *F*  This example identifies a Glucose lab test with a numeric expected returned value. There is no subcategory associate with this test, and the test specimen can come from multiple sources: blood, plasma, serum, serum/plasma, or unknown origin. This test specifies that the test subject was instructed to fast (*F*) prior to testing. |
| *Example 1.2* – **MS\_Test\_Name=** *GLUCOSE*, **Result\_Type**=*N*, **\_MS\_Test\_Sub\_Category**= <*null>*, **\_Specimen\_Source**=*BLOOD|PLASMA|SERUM|SR\_PLS|UNK*, **Fast\_Ind**= *R*  This example identifies a Glucose lab test with a numeric expected returned value. There is no subcategory associate with this test, and the test specimen can come from multiple sources: blood, plasma, serum, serum/plasma, or unknown origin. This test was random (R), and no specifications for fasting are reported. |
| *Example 1.3* – **MS\_Test\_Name=** *TROP\_T*, **Result\_Type**=*C*, **\_MS\_Test\_Sub\_Category**= <*null>*, **\_Specimen\_Source**=*BLOOD|PLASMA|SERUM|SR\_PLS|UNK*, **Fast\_Ind**= *X*  This example indicates a troponin T cardiac test with an expected character value returned. There is no subcategory associate with this test, and the test specimen can come from multiple sources: blood, plasma, serum, serum/plasma, or unknown origin. A fasting indicator is not applicable (X) to this test. |
| *Example 1.3* – **MS\_Test\_Name=** *TROP\_T*, **Result\_Type**=*N*, **\_MS\_Test\_Sub\_Category**= <*null>*, **\_Specimen\_Source**=*BLOOD|PLASMA|SERUM|SR\_PLS|UNK*, **Fast\_Ind**= *X*  This example indicates a troponin T cardiac test with an expected numeric value returned. There is no subcategory associate with this test, and the test specimen can come from multiple sources: blood, plasma, serum, serum/plasma, or unknown origin. A fasting indicator is not applicable (X) to this test. |

#### lkp\_lab\_result\_ranges

For select values of **MS\_Test\_Name** in the lkp\_lab\_test container, ranges for expected test values are defined. The worksheet is setup to resemble an input SAS data set to create formats, which it is used for in the QA Package labs module. Counts aggregated to the groupings defined by these formats will be used for internal SOC reference.

#### Lkp\_pre\_rxroute

This container contains the listing of valid values for populating the variable RxRoute in the SCDM prescribing table. Each row contains a unique route value with text description. The container is exported to create the SAS dataset *msoc.lkp\_pre\_rxroute*.

#### lkp\_pre\_rxdoseform

This container contains the listing of valid values for populating the variable RxDoseForm in the SCDM prescribing table. Each row contains a unique dosage form value with text description. The container is exported to create the SAS dataset *msoc.lkp\_pre\_rxdoseform*.

#### lkp\_pre\_rxdoseunit

This worksheet contains the listing of valid values for populating the variable RxDoseUnit in the SCDM prescribing table. Each row contains a unique dosage unit value with text description. The container is exported to create the SAS dataset *msoc.lkp\_pre\_rxdoseunit*.

#### lkp\_pvd\_specialty

This container contains the listing of valid values for populating the variable Specialty and its correlating Specialty\_CodeType value in the SCDM provider table. Each row contains a unique specialty value with text description based upon a subset of CMS’ specialty code list. The Speciality\_CodeType value correlates in length to 2-digit (2) or 10-digit (0) specialty codes. The container is exported to create the SAS dataset *msoc.lkp\_pvd\_specialty*.

#### lkp\_prm\_type

This container contains a hierarchical listing of Patient Reported Outcome/Measure (PRM) types, as well as associated sub-types. Each type and sub-type correlate with a free text description field. Currently, PRM tables are in early development phase and no output data sets are derived from this container.

#### lkp\_all\_saslength

This container contains two fields, Bytes and Capacity, which are used to delineate the maximum numeric integer (Capacity) that can be stored using a particular numeric SAS length value (Bytes). For example, a numeric SAS variable with length (Bytes) = 3 can hold all integers up to and including 8,192. The SAS dataset *msoc.lkp\_all\_saslength* is derived from this container.

#### lkp\_all\_flags

The contents of the lkp\_all\_flags container are used to create the lkp\_all\_flags SAS dataset as described in section 4.1. It contains a listing of all QA package flags which are referenced in each module that performs automated level 1, level 2, or level 3 checks. **Please Note:** This includes checks which are not directly invoked by the QA Package but are output to the master\_all\_flags.xlsx file as a reference for Sentinel Analysts.

Valid cell values are sourced from the \_list worksheet, and Excel verification utilizes drop down menus for the columns demarcated with “\*” below. When updating the lkp\_all\_flags container, the following information is required.

* **CheckID** columnis a three-digit identifier. **CheckID** must be in order, so when adding a new row, insert the row in the correct sequence.
* **Table1\*** columnis the table identifier for the table the check is run on (*all* means the check is run on all tables)
* **Table2\*** column is the table identifier of any second table when a cross table check is performed in level 2 and level 3 checks.
* **\_VarNum1\*** column contains the SAS variable as specified by its varnum in the evaluated dataset. A value of ‘00’ specifies no variable; the table is evaluated by the check. Special values indicate sets of variables being evaluated, such as all variables, missing variables, or compound key variables .
* **VarNum2 – VarNum4\*** columns contain any second, third, or fourth variables, specified by the SAS dataset varnum, which is included in the check.
* **\_TestNum\*\*** and **\_Result\_Type\*\*** columns contain values specific to lab tests. For non-lab checks, the value is *00*;
  + Values for test nums differentiate between all tests (*ta*), characterized tests (*tc*), characterized tests with required units (*tu*), characterized tests with only 1 result\_type (*tr*), and characterized tests without units (*tw*).
  + Result types are defined as all result types (*a*), character (*c*), or numeric (*n*)
* **AbortYN\*** column specifies a *Y* or *N* value to indicate whether SAS processing aborts if check is flagged.
* **FlagType\*** column specifies a *Fail* or *Warn* value to indicate whether the check causes the QA process to fail or report a warning if the check is flagged.
* **FlagYN**\*column specifies a *Y* or *N* value for flags found through manual check.
* **DatasetIn** field identifies the name of the temporary or permanent SAS dataset that is read in to evaluate whether any observations are flagged by the check.
* **DatasetOut** field is the resulting dataset containing the flagged observations caught by the check.
* **Lookup\_table** column contains values for more refine details of the check being done, if necessary. Including multiple lookup tables categorized by level (l1, l2) or topic (lab, MCV) functions as an organizational tool that allows for greater relational flexibility.
* **Var1, Var2, CheckID\_Description,** and **Temp** columns (formatted in gray background) are automatically filled via Excel functions which use values within this worksheet and other worksheets within the workbook.
  + **Var1**: using the Excel vlookup function,searches for the concatenated values of **Table1** and **\_VarNum1** in this worksheet and returns the correlating value of **Variable** (column T) in the \_List worksheet; NA if none is found.
  + **Var2:** using the Excel vlookup function,searches for the concatenated values of **Table2** and **VarNum2** in this worksheet and returns the correlating value of **Variable** (column T) in the \_List worksheet; NA if none is found.
  + **Temp:** uses the function Excel countif to return a count of the **CheckID** row value in the **CheckID** column.
  + **CheckID\_Description:** using the Excel Index function, returns corresponding conditional statements listed in the MasterKey worksheet.

*\*\*these are sources for derived variables in the final output dataset*

#### lkp\_all\_minmax

The lkp\_all\_minmax container is read in by the lookup table generator package to create the SAS dataset lkp\_all\_minmaxas described in section 4.1. This container provides for obtaining the *mindate* and *maxdate* from specific aggregated tables created during the QA Package process. The following fields need to be populated:

* **TabID** column should contain the lowercase 3-character table identifier.
* **Library** and **Dataset** columns contains the SAS libname and dataset name from which the minimum and maximum dates are derived.
* **DateVar** column defines the SAS variable from which the *mindate* and *maxdate* are derived.
* **CountVar** column contains the variable name used for aggregating count values. It indicates to the program module that the source data is stratified by other variables and needs further aggregation to acquire correct counts.
* **SumYN** column is a y/n value which informs the program as to whether the data containing the **CountVar** specified is stratified by extraneous variables and needs to be aggregated. Ultimately the **CountVar** and **SumYN** fields determine which **DateVar** value are counted, and how they are counted. Record counts aggregated by year-month are used to determine if a date reaches the percentage threshold for inclusion as a *mindate* or *maxdate*.
* **DPMinMax** column contains a y/n value, which ascertains whether the table’s *mindate* and *maxdate* informs the calculated overall **DPmindate** and **DPmaxdate** for the ETL being assessed.

|  |
| --- |
| **Example 1.0** – **TabID**=enr; **Library**=msoc; **Dataset**=enr\_l3\_enrmd\_ym; **DateVar**=yearmonth; **CountVar**=count; **SumYN**=n; **DPMinMax**=y  This configuration would specify that the Enrollment mindate and maxdate are derived from the *yearmonth* variable in msoc.enr\_l3\_enrmd\_ym. To check against an inclusion threshold, the variable *count* is used; it does not need to be summed to be used. The enrollment mindate and maxdate values are evaluated in the creation of the DP ETL’s min and max dates. |

#### control\_flow

The control\_flow container is used to create the control\_flow.csv input file as described in section 4.1. It is the driving force in the QA Package process, determining which modules are called in a predefined sequence. The control flow container contains the following columns which need to be populated.

* **module** field is the unique id for a module
* **execute\_flag** column takes a value of Y or N and indicates whether a module is run in the QA process.
* **sascode** column contains the name of the SAS program (without the .sas extension) that contains the QA package modular program being called.
* **cc\_table** field contains the standardized name of the SCDM table macro variable, where applicable, as expected in Common Components
* **seqno** is the ordered sequence number associated with each QA package module.
* **Unique\_id** field lists and single key identifier variable used by the table. It is used in the QA Package to acquire counts used for evaluating the accuracy of assigned numeric site-specific variable lengths.

|  |
| --- |
| **Example 1.1 - module**=l1, **execute\_flag**=Y, **sascode**=01.1\_scdm\_data\_qa\_review-level1, **cc\_table**=X, **seqno**=1, **unique\_id**=X  This configuration would run 01.1\_scdm\_data\_qa\_review-level1.sas program. The Level 1 module is run across multiple tables and thus has no specified cc\_table value. It is the first module run in the QA package sequence. |
| **Example 1.2 - module**=dis, **execute\_flag**=Y, **sascode**=04.1\_scdm\_data\_qa\_review-dispensing, **cc\_table**=DISTABLE, **seqno**=5, **unique\_id**=X  This configuration runs the 04.1\_scdm\_data\_qa\_review-dispensing.sas program; the DP’s dispensing table, referenced by the common components alias DISTABLE, is evaluated as the fifth module in the QA package sequence. |
| **Example 1.3 - module**=irx, **execute\_flag**=N, **sascode**=08.1\_scdm\_data\_qa\_review-inpatientpharmacy, **cc\_table**=IRXTABLE, **seqno**=13, **unique\_id**=RxID  This configuration specifies that the irx modular SAS program will not run. The IRXTABLE is an Inpatient table and is thus an optional table in the DP’s ETL process. If the IRXTABLE was utilized, SOC expects one row per RxID |

#### MacroVariables

This worksheet is intended for use by the developer only. It is not a table that exists in the QA package. It holds definitions for global utility macro variables.

#### MasterKey

This worksheet is intended for use by the developer only. It is not a table that exists in the QA package. The container documents the existing Master CheckID flag key at the time of package execution.