SENTINEL COMMON DATA MODEL

DATA QUALITY REVIEW AND CHARACTERIZATION PROCESS AND PROGRAMS

Program Package version: 4.1.0

Prepared by the Sentinel Operations Center
February 2018

The Sentinel System is sponsored by the U.S. Food and Drug Administration (FDA) to proactively monitor the safety of FDA-regulated medical products and complements other existing FDA safety surveillance capabilities. The Sentinel System is one piece of FDA's Sentinel Initiative, a long-term, multi-faceted effort to develop a national electronic system. Sentinel Collaborators include Data and Academic Partners that provide access to healthcare 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.

Table of Contents

Modification History

Version	Date	Modification	By
4.1.0	02/22/2018	- Minor bug fixes - Optimization of L1 module	Sentinel Operations Center
4.0.2	11/14/2017	- Minor bug fix	Sentinel Operations Center
4.0.1	09/2/2017	- Minor bug fixes	Sentinel Operations Center
4.0	04/27/2017	- Major updates	Sentinel Operations Center
3.3.4	2/16/2017	- Added/modified diagnosis datasets	Sentinel Operations Center
3.3.3	8/17/2016	- Added/modified core and lab datasets	Sentinel Operations Center
3.3.2	9/30/2015	- Corrected error in PatID exclusion in Labs module	Mini-Sentinel Operations Center
3.3.1	9/17/2015	- Corrected error in DEM module	Mini-Sentinel Operations Center
3.3	5/1/2015	- Redesign of labs module, addition of core checks	Mini-Sentinel Operations Center
3.2.4	9/23/2014	- Updated program for new RequestID compatibility, fixed bug	Mini-Sentinel Operations Center
3.2.3	9/2/2014	- Implemented minor bug fixes	Mini-Sentinel Operations Center
3.2.2	8/26/2014	- Implemented minor bug fixes	Mini-Sentinel Operations Center
3.2.1	7/25/2014	- Implemented minor bug fixes	Mini-Sentinel Operations Center
3.2	5/30/2014	- Integrated Common Components - Added MSCDM v4.0 compliance checks, minor bug fixes	Mini-Sentinel Operations Center
3.1.3	03/03/2014	- Implemented minor bug fixes	Mini-Sentinel Operations Center
3.1.2	09/25/2013	- Implemented minor bug fix	Mini-Sentinel Operations Center
3.1.1	09/16/2013	- Implemented minor bug fix	Mini-Sentinel Operations Center
3.1	09/05/2013	- Added/modified lab and vitals checks, fixed bugs	Mini-Sentinel Operations Center
3.0.1	03/01/2013	- Updated for UNIX compatibility	Mini-Sentinel Operations Center
3.0	11/12/2012	- Added PatID and EncounterID matching, enhanced valid date checks, fixed bugs - Added clinical data (labs and vitals) programs	Mini-Sentinel Operations Center
2.0	10/14/2010	- Added duplicate record checks, modified dataset names, fixed bugs	Mini-Sentinel Operations Center
1.0	09/20/2010	- Initial published version	Mini-Sentinel Operations Center

I. OVERVIEW

This document describes the version 4.1.0 program package used by the Sentinel Operations Center (SOC) for data quality assurance (QA) review and characterization of the Sentinel Distributed Database (SDD). To create the SDD, each Data Partner (DP) transforms local source data into Sentinel Common Data Model (SCDM) format. The SOC uses a set of data quality review and characterization programs to ensure that the SDD meets reasonable standards for data transformation consistency and quality. The version 4.1.0 QA program package queries SCDM version 6.0.2 tables.

II. DISTRIBUTED PROGRAMMING AND DATA QUALITY REVIEW CHECKS

To evaluate data characteristics and quality, SOC developed distributed code to query the content of SCDM-formatted tables. The distributed code generates aggregate output tables that help determine whether the data conform to SCDM specifications, maintain integrity across variables and across tables, and trend as expected over time. Output tables are designed to evaluate one or more data checks, i.e., pre-defined data quality measures or characterizations. Each data check is designated a “level 1,” “level 2,” or “level 3” data quality check depending on the complexity and assigned a “FlagID” for tracking and reporting purposes. A “FlagID” can represent a data characteristic or a data issue (see Section **IV.C.2.** for more information on FlagIDs).

Level 1 data checks review the completeness and content of each variable in each table to ensure that the required variables contain data and conform to the formats specified by the SCDM specifications. For each SCDM variable, level 1 data checks verify that data types, variable lengths, and SAS formats are correct and reported values are acceptable. For example, ensuring that the variable SEX in the *Demographic* table has a value of “F,” “M,” “A,” or “U” is a level 1 data check. Another example is ensuring that the variable MS_RESULT_C in the *Laboratory Result* table is only populated with values of “POSITIVE”, “NEGATIVE”, “BORDERLINE”, “UNDETERMINED”, or a RANGE: start|end unit (*i.e.,* “50|100 MG/ML”) for all laboratory tests.

Level 2 data checks assess the logical relationship and integrity of data values within a variable or between two or more variables within and between tables. For example, the SCDM requires that the variable ADMITTING_SOURCE in the *Encounter* table is populated only for inpatient and institutional encounters (*i.e.*, ENCTYPE= “IP” or “IS”). A level 2 data check would ensure that ADMITTING_SOURCE is populated only when ENCTYPE= “IP” or “IS”.

Level 3 data checks examine data distributions and trends over time, both within a Data Partner’s database (by examining output by year and year/month) and across a Data Partner’s databases (by comparing updated SCDM tables to previous versions of the tables). For example, a level 3 data check would ensure that there are no large, unexpected increases or decreases in diagnosis records over time.

III. NEW FEATURES/ENHANCEMENTS

A. EFFICIENCY

1. The new QA program package reduces the overall number of output files significantly, which reduces space and clarity of output.

B. CONSISTENT FLAG SCHEME

1. A new flag token structure now ensures consistency across all program modules. Each flag token can be linked to a flag description for easier interpretation.

C. FLAG INDICATORS

1. New flag indicators (i.e., "AbortYN", "FlagType") have also been introduced for each individual flag. This will help automate the program and provide clear indication of flag implication:

   – If AbortYN = "Y": the package will abort, errors will be reported in the log and all flags will be output to the "dplocal" folder for review by the Data Partner
   – If AbortYN = "N": the package will not abort, flags will be output to "dplocal" and/or "msoc" folder for review by the Data Partner and SOC

   – If FlagType = "Fail": the ETL cannot pass QA review until error is fixed
   – If FlagType = "Warn": the ETL may pass QA, but explanation or investigation could be warranted (i.e., more information is needed to determine QA pass/fail)

D. UPDATED MODULE FLOW

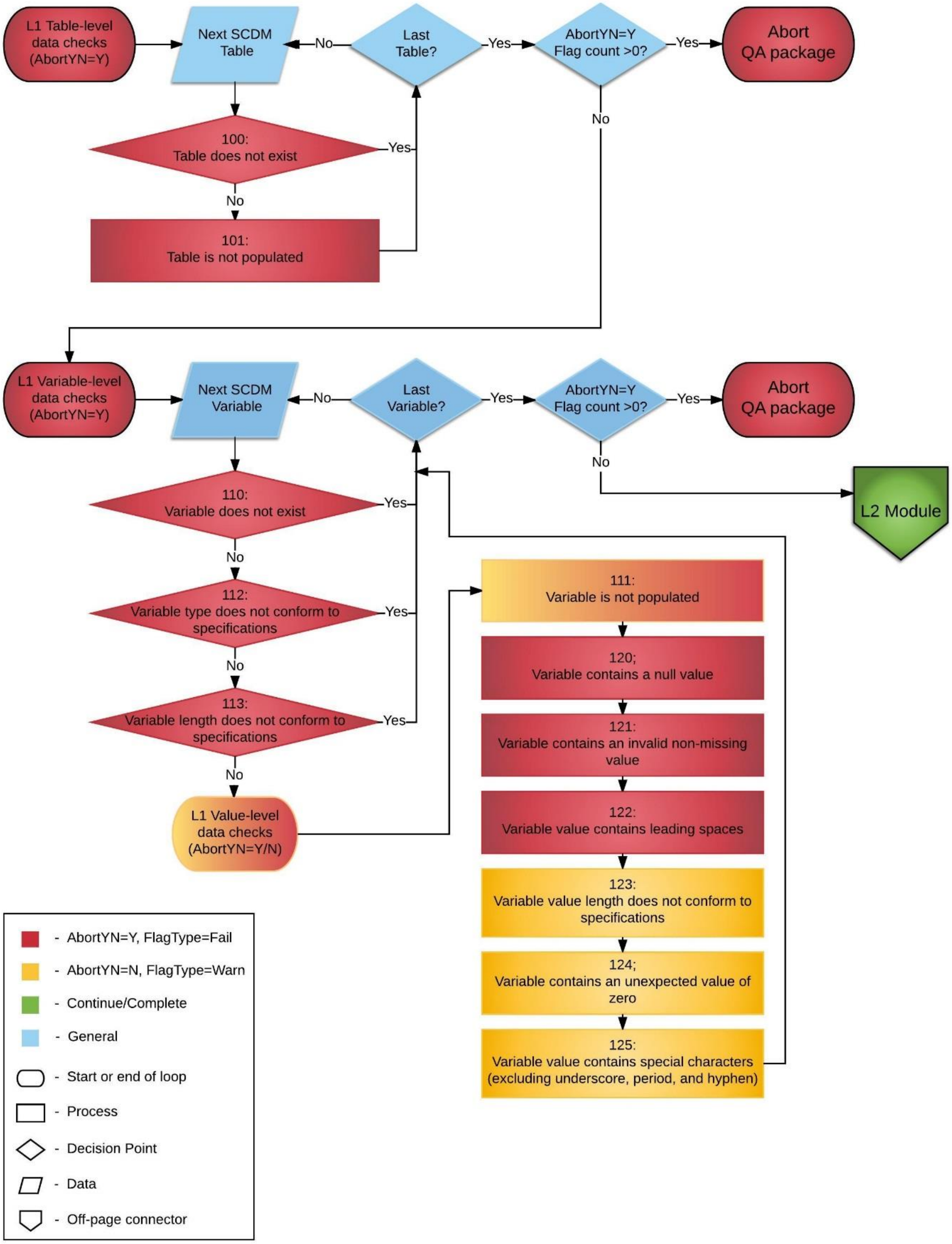
1. In the new QA program package, the module execution sequence has been reordered and module stop-gaps have been added as described below.

   a. All Level 1 checks for all tables are performed first and, if any major issues with the data are detected, the package will abort (**Figure 1**).
   b. All Level 2 checks are then performed in a logical sequence and abort at each step if "AbortYN" = Y for any flag in that step (**Figure 2**). The logical sequence is as follows:

      i. Perform critical intra-table checks that would cause downstream data integrity issues;
      ii. Perform critical cross-table checks for the same reasoning as above (this is the final place where the package may abort);
      iii. Continue to remaining Level 2 checks and all remaining modules, regardless of the resulting data flags.

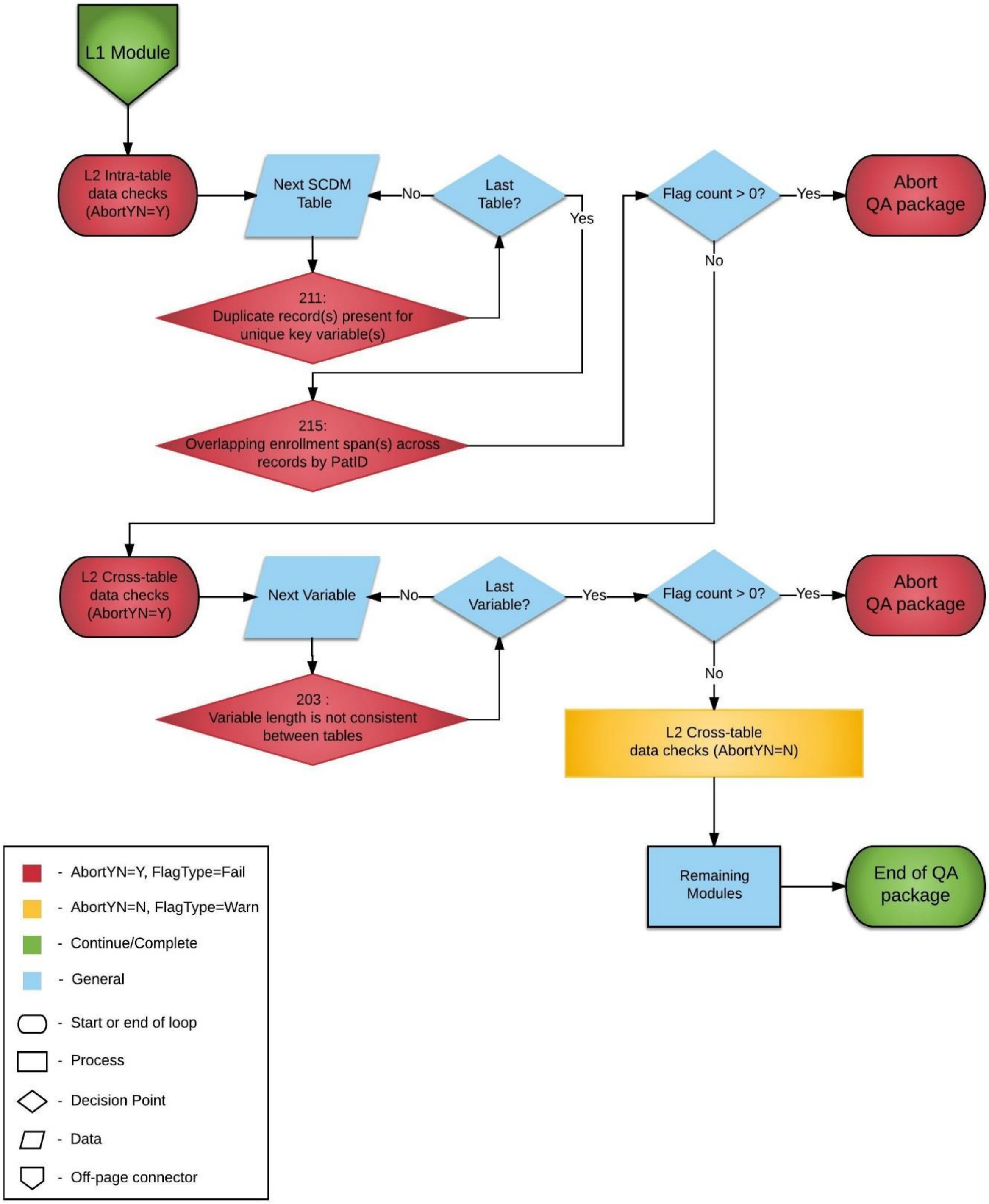
Note: Prior to step 1.b.iii., all resulting datasets are output to the "dplocal" folder. Only log, metadata, and signature files will be in the "msoc" subfolder, until all abort checks have been successfully executed. At that time, all appropriate datasets will be moved to the "msoc" subfolder. See Section **VII** for more information on "dplocal" and "msoc" output files.

Figure 1. Flowchart of Level 1 module abort logic[1]


L1 Table-level data checks (AbortYN=Y)
Next SCDM Table
No
Last Table?
Yes
AbortYN=Y Flag count >0?
Yes
Abort QA package
No
100: Table does not exist
Yes
No
101: Table is not populated
L1 Variable-level data checks (AbortYN=Y)
Next SCDM Variable
No
Last Variable?
Yes
AbortYN=Y Flag count >0?
Yes
Abort QA package
No
L2 Module
110: Variable does not exist
Yes
No
112: Variable type does not conform to specifications
Yes
No
113: Variable length does not conform to specifications
Yes
No
L1 Value-level data checks (AbortYN=Y/N)
111: Variable is not populated
120; Variable contains a null value
121: Variable contains an invalid non-missing value
122: Variable value contains leading spaces
123: Variable value length does not conform to specifications
124; Variable contains an unexpected value of zero
125: Variable value contains special characters (excluding underscore, period, and hyphen)
🟥 - AbortYN=Y, FlagType=Fail
🟨 - AbortYN=N, FlagType=Warn
🟩 - Continue/Complete
🟦 - General
- Start or end of loop
- Process
- Decision Point
- Data
- Off-page connector

[1] The rules are governed by two lookup tables, **lkp_all_l1** and **lkp_all_flags**, which are located in the “inputfiles” directory of the QA program package.

Figure 2. Flowchart of Level 2 module abort logic[2]


L1 Module
L2 Intra-table data checks (AbortYN=Y)
Next SCDM Table
No
Last Table?
Yes
Flag count > 0?
Yes
Abort QA package
No
211: Duplicate record(s) present for unique key variable(s)
215: Overlapping enrollment span(s) across records by PatID
L2 Cross-table data checks (AbortYN=Y)
Next Variable
No
Last Variable?
Yes
Flag count > 0?
Yes
Abort QA package
No
203 : Variable length is not consistent between tables
L2 Cross-table data checks (AbortYN=N)
Remaining Modules
End of QA package
🟥 - AbortYN=Y, FlagType=Fail
🟨 - AbortYN=N, FlagType=Warn
🟩 - Continue/Complete
🟦 - General
▢ - Start or end of loop
□ - Process
◇ - Decision Point
▱ - Data
⬠ - Off-page connector

[2] Please note that flags in figures 1 and 2 may change over time.

IV. PROGRAM PACKAGE

A. FOLDER STRUCTURE

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

1. **dplocal**

   The subfolder containing output generated by the request that should remain with the DP (and may be used to facilitate follow-up queries).

2. **inputfiles**

   The subfolder containing all input files and lookup tables needed to execute a request. Input files contain information on what tables should be output and the type of analyses conducted on the variables in each table. Input files are created for each run of the QA program package by the SOC DMQA team.

3. **msoc**

   The subfolder containing output generated by the local SAS program. This will be used by SOC for post-QA processing.

4. **sasprograms**

   The subfolder containing the master SAS program that must be edited and then executed by the SOC analyst.

B. SAS PROGRAM MODULES

Discrete program functions should be contained in separate macros, to facilitate use by multiple programmers simultaneously, and for ease of future modification. The program package includes the following SAS modules:

1. **sasprograms/00.0_mscdm_data_qa_review_master_file.sas**

   This master program requires editing by the Data Partner to identify the location of the SCDM tables in staging, as well as the location of the Common Components program. It is the only program that is edited and executed by the partner.

2. **inputfiles/00.0_mscdm_control_flow.sas**

   This program module selectively and sequentially executes the QA program modules.

3. **inputfiles/00.1_mscdm_standard_macros.sas**

   Contains macros used across QA program modules.

4. **inputfiles/00.2_mscdm _formats.sas**

   Contains standard formats used across QA program modules.

5. **inputfiles/00.3_mscdm_sas_log_checker_directory_cc.sas**

   Checks all program logs and summarizes notes, warnings, and errors in an output PDF file after all modules have completed.

6. **inputfiles/00.4_mscdm_qasignaturerequest.sas**

   Creates a final signature file (msoc.alltable_signature) that summarizes metadata from individual module-level signature files.

7. **inputfiles/01.1_mscdm_data_qa_review-level1.sas**

   Queries applicable SCDM tables to perform level 1 data checks and creates all l1 output datasets. The data checks included in the module check compliance with the current SCDM specifications (*e.g.*, appropriate length, type, and format). Reference **Figure 1** above.

8. **inputfiles/01.2_mscdm_data_qa_review-level2.sas**

   Queries applicable SCDM tables to perform all level 2 abort checks, as well as L2 cross-table checks and output datasets. Reference **Figure 2** above.

9. **inputfiles/02.1_mscdm_data_qa_review-enrollment.sas**

   Queries the *Enrollment* table, and outputs L2 and L3 datasets containing information on medical and drug coverage indicators, enrollment start, end, enrollment duration, and chart availability.

10. **inputfiles/03.1_mscdm_data_qa_review-demographic.sas**

    Queries the *Demographic* table, and outputs L2 and L3 datasets containing information on age, sex, race, and zip code.

11. **inputfiles/04.1_mscdm_data_qa_review-dispensing.sas**

    Queries the number of members and records in the *Dispensing* table, and outputs information on dispensing date, dispensings over time, dispensings per member over time, days supply and dispensed amount.

12. **inputfiles/05.1_mscdm_data_qa_review-encounter.sas**

    Queries the number of members, records, encounters, and providers in the *Encounter* table, and outputs information on admission and discharge date, encounters over time, encounter type, length of stay, facility location, admitting source, discharge status and disposition, DRG and DRG type, and number of encounters per member.

13. **inputfiles/05.2_mscdm_data_qa_review-diagnosis.sas**

    Queries the number of members, records, encounters, and providers in the *Diagnosis* table, and outputs information on encounter type, admission date, diagnosis code and type, principal diagnosis indicators, number of diagnoses per encounter, and number of diagnoses over time.

14. **inputfiles/05.3_mscdm_data_qa_review-procedure.sas**

    Queries the number of members, records, encounters, and providers in the *Procedure* table, and outputs information on encounter type, admission date, procedure code and type, number of procedures per encounter, and number of procedures over time.

15. **inputfiles/06.1_mscdm_data_qa_review-death.sas**

This module is only executed by Data Partners who have death data available. Queries the number of members and records in the *Death* table, and outputs information on the source of and confidence in death information, number of deaths over time, and if the death date has been imputed.

16. **inputfiles/06.2_mscdm_data_qa_review-causeofdeath.sas**

    This module is only executed by Data Partners who have cause of death data available. Queries the number of members and records in the *Cause of Death* table, and outputs information on cause of death codes and cause type, and source of and confidence in cause of death information.

17. **inputfiles/07.1_mscdm_data_qa_review-labs.sas**

    This module is only executed by Data Partners who have laboratory data available. This program queries the number of members and records in the *Laboratory Result* table and outputs information on lab tests included, result values, units, and available dates.

18. **inputfiles/99.1_mscdm_data_qa_review-minmax_dates.sas**

    Queries SCDM tables and outputs minimum and maximum dates per table (as applicable), and calculates DP Min (calculated as the maximum of the Min Dates) and DP Max (calculated as the minimum of the Max dates).

19. **inputfiles/99.2_mscdm_data_qa_review-level3.sas**

    Creates Level 3 cross-table and age-related datasets from intermediate table-level datasets, and performs many “housekeeping” activities, such as moving specific files from “dplocal” to “msoc”, and bulk addition of DPID and SITEID variables to “msoc” datasets.

C. LOOKUP FILES

A set of lookup tables will be included in the ‘/inputfiles’ directory of the QA program package to allow easy modifications of value sets and error codes.

1. **control_flow.sas7bdat**

   Used by control flow module to selectively and sequentially execute QA modules. Used in the L1 and L2 modules to selectively execute data checks by SCDM table.

2. **lkp_all_flags.sas7bdat**

   This lookup table provides a list of DMQA-assigned error codes (*FlagID*) and descriptions (*Flag_Descr*), and the name of the output dataset used to evaluate checks associated with the error code.

   Each error code (FlagID) is comprised of five meaningful tokens:

   1. Table(s) Abbreviation (*Table*): Abbreviation that indicates SCDM table(s) queried for the data check
   2. Check Level (*Level*): Level of data check; i.e., the type of quality assurance check performed (1=basic, single variable model compliance; 2=cross-variable and cross-table compliance checks; 3= temporal trends within and across database versions)
   3. Variable identifier (*VarID*): Unique variable identifier

4. Test Identifier (*TestID*): Unique MS_Test_Name value identifier concatenated with Result_type (e.g. 01-N)
5. Check Identifier (*CheckID*): Unique data check identifier

Each of these tokens is also included as a separate variable in the lookup table.

Data checks for which the value of the variable *FlagYN* = "Y" indicate that the check is performed automatically and output to the flags output dataset. Level 1 and 2 error codes that do not pass automatic evaluation (i.e., have a count of one or more records that meet the error code description) will get written a flags dataset.

AbortYN = "Y" is used to indicate checks that must pass in order to complete the QA module successfully. Data checks for which the value of the variable *FlagType*= "Fail" indicate that the check must pass in order to pass QA review.

3. **lkp_all_l1.sas7bdat**

   This lookup table provides information on all variable attributes by SCDM table in order to perform Level 1 model compliance checks. For each variable in the table, the variable identifier, required type (numeric or character), and required length are specified. An indicator of whether a variable is allowed to have a missing/blank value is also included, as well as a list of all allowable variable values (as applicable).

4. **lkp_all_minmax.sas7bdat**

   This lookup table provides table-level information such as source table and variable names, as well as inclusion status in order to calculate minimum and maximum dates of data completeness.

5. **lkp_dem_zip.sas7bdat**

   Zip code lookup that links valid 5-digit zip codes with the associated state. SAS creates a zip file quarterly and is the source of this data.

6. **lkp_dia_l2.sas7bdat**

   This lookup table provides a comprehensive list of all allowable cross variable value combinations to be used for the *Diagnosis* table module Level 2 data checks.

7. **lkp_enc_l2.sas7bdat**

   This lookup table provides a comprehensive list of all allowable cross variable value combinations to be used for the *Encounter* table module Level 2 data checks.

8. **lkp_lab_l2.sas7bdat**

   This lookup table provides a comprehensive list of all allowable combinations for the following variables in the *Laboratory Result* table: MS_Test_Name, Result_type (numeric or character), MS_Test_Sub_Category, Fast_ind, Specimen_Source, and LOINC. To be used for Level 2 error checks.

9. **lkp_lab_l2_nc.sas7bdat**

   Allowable *Laboratory Result* table lab test result, modifier, and unit combinations. This lookup table maps the acceptable MS_Result_C values to the Modifier values and MS_Result_unit values, as appropriate. Only characterized laboratory tests are included in this lookup table.

10. **lkp_lab_result_ranges.sas7bdat**

    Expected *Laboratory Result* table lab test ranges. This lookup table defines the expected result value ranges for lab tests with numeric results in the *Laboratory Result* table. Only MS_Test_Name values where Result_Type= "N" and lkp_lab_test.Characterized="Y" are included in this lookup table.

11. **lkp_lab_test.sas7bdat**

    Lab Test/Test type relationship. This lookup table provides a comprehensive list of all laboratory tests included in the *Laboratory Result* table. It includes laboratory test IDs and associated MS_Test_Name, Result_type, whether a unit is required, and whether a test has been characterized or not. To be used for Level 2 error checks.

12. **lkp_pro_l2.sas7bdat**

    This lookup table provides a comprehensive list of all allowable cross variable value combinations to be used for the *Procedure* table module Level 2 data checks.

V. PROGRAM EXECUTION

A. MASTER PROGRAM INPUT TO BE COMPLETED BY SOC

Table 1 below defines the variables that must be initialized by the SOC to execute the QA program package (*i.e.*, defined by the SOC before execution of the programs). Please note that these values cannot be left blank. These inputs are unique to each request and/or Data Partner.

Table 1. Master Program Variable Definitions to be completed by SOC

Field Name	Description
ReqETL	Request ETL number
MSProjID	Project ID
MSWPType	Workplan Type
MSWPID	Workplan ID
MSDPID	Unique Data Partner ID
MSVerID	Version ID

B. MASTER PROGRAM INPUT TO BE COMPLETED BY DP

Table 2 below defines the variables that must be initialized by the end user to execute the QA program package (*i.e.*, defined by the Data Partner before execution of the programs). Please note that these values cannot be left blank. Each Data Partner is required to enter user inputs at the beginning of the applicable data quality program. These inputs are unique to each Data Partner.

Table 2. Master Program Variable Definitions to be completed by Data Partner

Field Name	Description
MSCC	Path to directory containing the Common Components file (ms_common_components.sas)
Evaluate_MSCDM	Path to directory containing the SCDM datasets pending QA review

VI. PROGRAM ALGORITHMS/LOGIC

A. DEFINITION OF ENROLLMENT SPAN COMPARISONS

Table 3. Definitions and examples of enrollment date range relationships by PatID

Definition	Illustration	Enr_Start	Enr_End	DMQA Action
Disjoint: No conflict or overlap	———	**01/01/2015**	**03/31/2015**	Pass
	———	**05/01/2015**	**08/31/2015**	
Consecutive: Two date ranges are consecutive	———	**01/01/2015**	03/31/2015	Warn
	———	04/01/2015	**08/31/2015**	
Overlap: Two date ranges overlap over a range	———	**01/01/2015**	03/31/2015	Fail
	——————	02/01/2015	**08/31/2015**	
Duplicate: Two date ranges are identical	———————	**01/01/2015**	**08/31/2015**	Fail
	———————	01/01/2015	08/31/2015	
Subset: One date range is a subset of another	———————	**01/01/2015**	**08/31/2015**	Fail
	——	02/01/2015	03/31/1015	

B. MINIMUM AND MAXIMUM DATES OF DATA COMPLETENESS

Minimum and Maximum dates (min/maxdate) of data completeness are created by this package for all SDCM tables containing at least one date variable, as defined in the input file *lkp_all_minmax.sas7bdat*.

The mindate is calculated by determining the earliest year-month (e.g. 2010-01) with a record count within an 80% threshold of the next month (e.g. 2010-02) and then assigning the first day of the month to create a SAS date, formatted as YYYY-MM-DD (e.g. 2010-01-01).

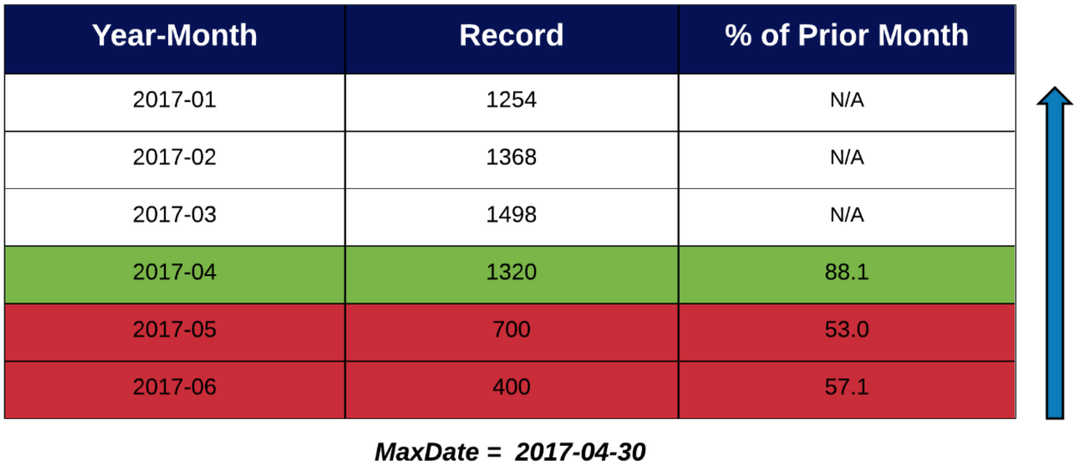
The maxdate is calculated by determining the latest year-month (e.g. 2017-10) with a record count within an 80% threshold of the prior month (e.g. 2017-09) and then assigning the last day of the month to create a SAS date, formatted as YYYY-MM-DD (e.g. 2017-10-31).

Overall min/maxdates are then created, based on the SCDM table min/maxdates, as defined in the input file *lkp_all_minmax.sas7bdat*. The overall mindate is calculated by determining the latest mindate (i.e. the "maximum of the minimum"). The overall maxdate is calculated by determining the earliest maxdate (i.e. the "minimum of the maximum").

These dates are stored in a SAS dataset (msoc.minmax_dates). The overall min/maxdates associated with the latest production ETL at each DP site will be used by Common-Components (CC) to populate the global macro variables &mindate and &maxdate for distributed request packages.

It should be noted that the min/maxdate algorithm may not work well with all types of date distributions (e.g., a distribution with a large drop proceeded or followed by a long, flat tail of many months).

Figure 3. Example of Maximum date of data completeness algorithm[3]



Year-Month	Record	% of Prior Month
2017-01	1254	N/A
2017-02	1368	N/A
2017-03	1498	N/A
2017-04	1320	88.1
2017-05	700	53.0
2017-06	400	57.1

MaxDate = 2017-04-30

[3] When there are at least two consecutive months at the tail end of the distribution with relatively low counts, the algorithm may sometimes pick a month with incomplete data. For example, if the month 2017-06 had a count of "600" instead of "400", it would meet the 80% threshold and be incorrectly chosen as the max date.

C. AGE CALCULATION

Age in years (age_years) is calculated using the Kreuter method using the date of birth variable (DEM.birth_date) and the new Overall MaxDate macro variable (&dp_maxdate) calculated for the ETL under review.

The following equation (first proposed by William Kreuter) measures age in whole years. It counts the months between the two dates, subtracts one month if the day boundary has not been crossed for the last month, and then converts months to years and reports it as an integer.

```
Age_years= floor((intck('month',birth_date,&DP_MaxDate.)-
           (day(&DP_MaxDate.)<day(birth_date)))/12)
```

D. AGE GROUP CATEGORIES

Age in years will be summarized based on the following categories:

```
"00. Missing"
"00. Negative"
"01. 0-1 yrs"
"02. 2-4 yrs"
"03. 5-9 yrs"
"04. 10-14 yrs"
"05. 15-18 yrs"
"06. 19-21 yrs"
"07. 22-44 yrs"
"08. 45-64 yrs"
"09. 65-74 yrs"
"10. 75+ yrs"
```

E. MODULE-LEVEL EXECUTION SIGNATURE FILES

Individual module-level signature files (msoc.{module}_signature) containing metadata and basic benchmarking statistics are created after each module executes.

The table below describes the contents of the enr_signature file:

Variable	Description	Source/Derivation	Example Values
DPID	2 letter Data Partner ID	Common Components &DPID	MS
SiteID	1-4 letter Site ID	Common Components &SITEID	OC
MSReqID	Request ID	Master program &MSREQID	soc_qar_v4_msoc_b05
MSProjID	Project ID	Master program &MSPROJID	soc
MSWPType	Workplan Type	Master program &MSWPTYPE	qar
MSWPID	Workplan ID	Master program &MSWPID	v4
MSDPID	Unique DP Site Identifier	Master program &MSDPID	msoc
MSVerID	Request Version ID	Master program &MSVERID	b05
QAVer	QA program package version	Master program &QAVER	4.1.0
SCDMVer	Current SCDM version	Master program &SCDMVER	6.0.2
Module	QA program package Module	Control flow &MODULE	enr
OSABBR	Operating System Environment	SAS automatic macro variable &SYSSCP	WIN
OSNAME	Operating System Name	SAS automatic macro variable &SYSSCPL	X64_7PRO
SASVersion	SAS version (short)	SAS automatic macro variable &SYSVER	9.4
SASVersionLong	SAS version (long)	SAS automatic macro variable &SYSVLONG	9.04.01M3P062415
RunType	SAS execution mode*	SAS automatic macro variable &SYSENV	FORE
NCPU	Number of CPU	SAS automatic macro variable &SYSNCPU	4
StartTime	Module start time	Standard macros %TIMESTAMP	12SEP2017:11:32:48.00
StopTime	Module end time	Standard macros %TIMESTAMP	12SEP2017:11:32:58.40
Seconds	Module run time in seconds	Standard macros %TIMEREPORT	10
RunTime	Module run time	Standard macros %SIGNATURE_END	0 h 0 m 10 s

* FORE=Interactive, BACK=Batch

F. REQUEST-LEVEL EXECUTION SIGNATURE FILES

A single signature file (msoc.alltable_signature) contains request-level metadata and basic benchmarking statistics after all modules have completed, and summarizes data from all module-level signature requests.

The table below describes the contents of the alltable_signature file:

Variable	Description	Source/Derivation	Example Values
DPID	2 letter Data Partner ID	Common Components &DPID	MS
SiteID	1-4 letter Site ID	Common Components &SITEID	OC
MSReqID	Request ID	Master program &MSREQID	soc_qar_v4_msoc_b05
MSProjID	Project ID	Master program &MSPROJID	soc
MSWPType	Workplan Type	Master program &MSWPTYPE	qar
MSWPID	Workplan ID	Master program &MSWPID	v4
MSDPID	Unique DP Site Identifier	Master program &MSDPID	msoc
MSVerID	Request Version ID	Master program &MSVERID	b05
QAVer	QA program package version	Master program &QAVER	4.1.0
SCDMVer	Current SCDM version	Master program &SCDMVER	6.0.2
OSABBR	Operating System Environment	SAS automatic macro variable &SYSSCP	WIN
OSNAME	Operating System Name	SAS automatic macro variable &SYSSCPL	X64_7PRO
SASVersion	SAS version (short)	SAS automatic macro variable &SYSVER	9.4
SASVersionLong	SAS version (long)	SAS automatic macro variable &SYSVLONG	9.04.01M3P062415
RunType	SAS execution mode	SAS automatic macro variable &SYSENV	FORE
NCPU	Number of CPU	SAS automatic macro variable &SYSNCPU	4
TotalRequestTime	QA program package run time	Signature request module	0 h 6 m 27 s

VII. APPENDICES

Execution of all modules of the QA program package generates the output files in Appendix A and Appendix B.

A. **APPENDIX A: PROGRAM PACKAGE OUTPUT IN “DPLOCAL”**

Reference Appendix A for a list of “dplocal” datasets.

B. **APPENDIX B: PROGRAM PACKAGE OUTPUT IN “MSOC”**

Reference Appendix B for a data dictionary containing “msoc” datasets.