

MDS 3.0 Data Submission Specifications Overview

Version 1.13.2

1 Introduction

Version 3.0 of the Minimum Data Set (MDS) was implemented on October 1, 2010. New data submission specifications have been developed to support the new version of the MDS. This document, as well as a set of supporting files and documents, details these new data submission requirements.

Previous users of the MDS data submission specifications should note that the structure of the data specification reports and of the data submission files themselves have changed substantially from MDS 2.0. All users of the specifications are strongly urged to read through this document carefully.

2 Version History

The table below summarizes the versions of the data submission specifications that have been released along with their effective dates.

Table 1: Data Submission Specifications Version History

Data Specs Version	Effective Start Date	Effective End Date
1.00	10/01/2010	03/31/2011
1.01	04/01/2011	09/30/2011
1.02	10/01/2011	03/31/2012
1.10	04/01/2012	09/30/2012
1.11	10/01/2012	05/18/2013
1.12	05/19/2013	09/30/2013
1.13	10/01/2013	none

Version 1.13.2 of the data submission specifications incorporates all of the changes that were described in the latest errata document for Version 1.13.1. That errata document was dated July 30, 2013 and contained 17 issues.

All changes associated with Version 1.13.2 are identified in the *Item Change Report* and in the *Edit Change Report*.

3 Version Implementation

Each published version of the data specifications has a version number which is formatted as N.NN.NN (e.g., "1.00.1". The first portion of the version number (e.g., "1.00") is referred to as the major version number, and the last portion (e.g., ".1") is referred to as the minor version number. The major version number is incremented whenever there is a substantive change to the data specifications that requires software changes. Minor version numbers are incremented when minor changes or corrections are made to a major version.

When a new major version of the data specs is published, it will have an associated starting effective date. The version that is in effect when the new specs are published will have an ending effective date on the day before the new version takes effect.

For example, version 1.00 was the initial version of the MDS 3.0 data specs. It had a starting effective date of 10/1/2010. In January, 2011, the first new major version of the data specs, version 1.01, was published. Version 1.01 has a starting effective date of 4/1/2011. Version 1.00 therefore ends on the previous day: 3/31/2011.

The data specs effective dates, in conjunction with the target date for a submitted record, determine which version of the data specs applies to the record. The target date for a submitted record is defined as follows:

- a) If A0310F is equal to [01], then the target date is equal to A1600 (entry date).
- b) If A0310F is equal to [10,11,12], then the target date is equal to A2000 (discharge date).
- c) If A0310F is equal to [99], then the target date is equal to A2300 (assessment reference date).

When a submitted record is validated by the Assessment Submission and Processing (ASAP) system, its target date is evaluated and is used to load the appropriate set of edits for the specifications in effect on the target date. If the submitted record does not conform to those edits, the appropriate warnings or error messages will be issued and the record may be rejected. Using V1.00 and V1.01 as examples, if the target date of a submitted record is 3/31/2011, then it must conform to V1.00. If the target date of a submitted record is 4/1/2011, then it must conform to V1.01.

Once a new version of the data specs takes effect, data submission software will typically have to handle records from the previous version (or versions) and the new version. That software must therefore calculate the target date for each record, determine which version of the specs applies, and use those specs to validate the record prior to submission. Failure to do this may result in warnings, fatal errors, or unexpected results.

For example, suppose that a new MDS 3.0 item is defined and activated in a new version of the data specs. If that new item is submitted for a record with a target date that precedes the new version of the specs, then that new item will be ignored. It will be ignored because the new item is unknown to the data specs that are in effect and the ASAP system ignores unknown items. If the new item is omitted from a record with a target date that is on or after the effective date of the new specs, then a fatal error will occur and the record will be rejected. This will occur because the ASAP system will apply the new version of the specs and will see that the new item was not submitted when it was required.

Note that the appropriate data specs version is determined regardless of the submission date. Thus, if a record is submitted in 2012 with a target date of 10/1/2010, version 1.00 will apply. It should also be noted that if a record is being modified using Section X, then the version of the data specs is determined using the target date of the modified record, just as if it was a new record.

When submitting a record (new record, modification, or inactivation), the SPEC_VRSN_CD field in the control section must be included. The SPEC_VRSN_CD is an informational item which indicates the version of the data specs that was used to create the record. The allowable values of SPEC_VRSN_CD correspond to the major version numbers that have been published, and the submitted value should match one of these values. If it does not, a warning will be issued. Note that if the value submitted does not match the version that is in effect based upon the target date of the record, no warning will be issued. Furthermore, the value submitted does not affect or control in any way the version of the data specs that is applied to the submitted record. As explained above, this is controlled solely by the target date of the submitted record.

On rare occasions, a change to the data specs may take effect based upon submission date rather than target date. When this occurs, it will be noted in the data specs.

4 Overview of Major Changes from MDS 2.0

A comparison of the MDS 3.0 with the MDS 2.0 will quickly reveal that many assessment items have been added, deleted, and revised. In addition, a number of structural changes have been made to the item set and to the data specifications themselves, as described below.

- **Item labels.** A new scheme has been introduced to label MDS items. This new scheme is intended to be more straightforward than the scheme used on MDS 2.0 and to allow more flexibility if new items are added in the future. The first letter of each item name reflects the section the item belongs to (e.g., “A” for items in Section A, “B” for items in Section B, and so on). Following this, four digits are used to indicate the item’s number within the section (e.g., A0100, A0200, etc.). Gaps have been intentionally left between these numbers so that additional items can be added at a later time, if necessary.

As on MDS 2.0, some items contain one, two, or three levels of sub-items that subdivide the item into multiple questions or that provide response options. First level sub-items are designated with a letter (e.g., A0100A, A0100B, and A0100C). Second level sub-items are designated with a number following the letter (e.g., D0500A1, D0500A2, D0500B1, D0500B2, etc.). Third level sub-items are designated with another letter (e.g., M0400B3A, M0400B3B).

An additional labeling convention has been applied to checklist items. As on MDS 2.0, checklist items are groups of “check all that apply” items that may be either checked or not and that have a “none of the above” item associated with them. Under MDS 3.0, the “none of the above” item is designated with the letter Z. For example, memory/recall ability checklist contains items C0900A through C0900D, plus C0900Z (none of the above). Use of the “Z” designation allows easy addition of checklist items to the group, if necessary in the future.

The convention used on the item set itself and in the data specifications is to always use upper case letters in the item labels.

- **Submission file format.** With MDS 3.0, there will be a major change in the submission file format. MDS 2.0 utilized fixed-format ASCII submission files with different submission record types (header, body, and trailer records). Each submission file could contain records for one or more assessments. MDS 3.0 will use files in Extensible Markup Language (XML). Each XML submission file must contain data for one and only one assessment. The XML tags that are used to identify the data for each item will correspond to the item labels described above. Thus, if an assessment has a value of “2” (married) for item A1200 (marital status), the tag would look like this:

```
<A1200>2</A1200>
```

The XML file structure will be described in more detail in a later section of this document.

- **Fixed file format.** As noted above, nursing home and swing bed providers will use XML files to submit data to CMS. However, the data specs also define a fixed format file layout which will be used in other circumstances, such as calling the RUG IV grouper DLL. For example, CMS will use the fixed file format for data extracts, such as those that will be used to periodically extract data which is sent to individual States. The fixed-format file layout will also be used to pass data to utilities (e.g., RUG groupers) that are provided by CMS.
- **Item Groups.** The data specifications contain a new concept (referred to as the “item group”) which can have the following values: “control”, “assessment”, “calculated”, and “filler”. Assessment items are simply those that are contained in the published item set, as well as State items in Section S. Control items are the items that previously would have been on the MDS 2.0 header record as well as those items that preceded and followed the assessment items in the MDS 2.0 body record (e.g., the production vs. test flag, the software vendor’s tax ID, etc.). Calculated items are those items that are calculated by the Assessment Submission and Processing (ASAP) system (e.g., RUG value, Urban/Rural code, resident age, etc.). Finally, filler items are reserved for future use on the fixed-format record.
- **Nursing home and swing bed providers.** Under MDS 2.0, slightly different versions of the assessment were created for nursing home and swing bed providers. These assessments were submitted to two separate data systems. MDS 3.0 will be used for both types of providers. A new item (A0200) is used to identify the type of provider that is submitting the assessment. A

single submission system, the ASAP system will collect assessments from both nursing home and swing bed providers.

- **Item subset codes.** Under MDS 2.0, assessment record type codes (REC_TYPE) were defined based upon the reason for assessment (RFA) items (AA8a and AA8b). The set of possible record types was smaller than the set of all possible combinations of AA8a and AA8b. This led to some simplification in specifying which items were active on the various types of assessment. A parallel concept will be used for MDS 3.0 and has been renamed “Item Subset Code” (ISC). This new name reflects the fact that the entire set of MDS 3.0 items represents the entire superset of assessment items that can be included on **any** type of assessment. **Subsets** of these items are included on various types of records (e.g., comprehensive, quarterly, discharge, entry, 5-day, etc.). ISCs have been defined based on the RFA items and are presented in a later section of this document. Please note that the RFA items (A0200, A0310A, A0310B, A0310C, A0310D, and A0310F) have been revised for MDS 3.0 and that the ISCs differ from the REC_TYPEs that were defined for MDS 2.0. Item subset codes and the reasons for assessment are a key part of the data specifications and are explained in a later section of this document.
- **Active, inactive, and state optional items.** MDS 3.0 has retained the concept of active, inactive, and state-optional items. The ISC will control the set of items that are active for a particular type of assessment. Only active items plus any state-optional items selected by the state in which the provider resides will need to be included in the XML submission file. XML elements for any inactive items should be excluded from the submission file.
- **Submission requirement items.** The MDS 2.0 body record contained a field call SUB_REQ that encoded submission requirement information for the record. This field indicated whether the record was being submitted to satisfy Federal or State requirements, or whether it was required by neither authority. This item has been made a part of the item set (see item A0410, Submission Requirement).
- **Section X.** On MDS 2.0, correction and inactivation procedures were implemented after the rollout of MDS 2.0 and the fields required to perform corrections and inactivations were an addition to the assessment (Section AT) and not a formal part of the assessment itself. On MDS 3.0, the items necessary to perform these transactions have been added to the item set and are contained in a new section called Section X.
- **IDs assigned to edits.** The data specifications have been completely rewritten. Perhaps the biggest improvement is that each edit (formatting, consistency, skip pattern, etc.) has been assigned a unique ID. If an edit applies to more than one item, the detailed data specifications report lists the edit ID along with the text of the edit so that it is unambiguously clear that the same edit applies to a set of items. In addition, each edit is systematically listed with every item that is included in the edit. A new report has been developed that contains an unduplicated list of edits along with a list of the items to which each edit applies. This report should assist developers in creating and validating their software. Finally, the edit IDs that are used in the data specifications will be used on the ASAP system feedback reports to assist those who wish to reconcile errors and warnings from the two sources.
- **LOINC codes.** In order to promote the use of electronic health records and standardized nomenclature systems, CMS will support the optional submission of LOINC codes that are associated with MDS items and with the each of the responses to those items. MDS software that supports electronic health records and standardized transmission protocols, such as HL7 (<http://www.hl7.org>), should benefit from the assignment of LOINC codes to MDS items and responses. At this time, LOINC codes have not been assigned, but when they are available relevant LOINC codes will be displayed on future data specifications reports.

5 Data Specifications Files

Two sets of files are included in the data specifications. The first set consists of reports and documentation that describe the data specifications. The second set is based upon the data dictionary that was used to generate the data specifications. This latter set of files will be useful to software developers. Note that in the file names below, *n.nn.r* stands for the version and revision number associated with the data specifications. The *n.nn* portion represents the version number, while *.r* represents the revision number. For example, **1.00.0** would be the initial release of Version 1.00. The first revision would be 1.00.1, the second would be 1.00.2, etc. In addition, the file names for draft versions of the documents will contain the word “draft” after the version number.

5.1 Reports and Documentation

- **MDS 3.0 data specs overview (vn.nn.r).pdf** The current document.
- **Data specs report (vn.nn.r).pdf** This report contains detailed data specifications for every item in the data set.
- **Undup edits report by ID (vn.nn.r).pdf** This report contains an unduplicated list of all edits (formatting rules, consistency checks, etc.) that apply to the item set. It is sorted by the edit ID number.
- **Item change report (vn.nn.r).pdf** This report lists changes that have been made to items or item responses since the previous release of the data specs. This report will not be produced for the initial release of the data specs, but will be included in subsequent releases.
- **Edit change report (vn.nn.r).pdf** This report lists changes that have been made to edits since the previous release of the data specs. This report will not be produced for the initial release of the data specs, but will be included in subsequent releases.
- **HTML data specs (vn.nn.r).zip** This zip file contains a set of HTML files that display the same information as is in the detailed data specs document. To use these files, unzip them to an empty folder and use a browser to open the file called INDEX.HTML. This will open a two-panel window. The left-hand panel can be used to navigate a list of the MDS items or of the MDS edits. When an item or edit is selected, the right-hand panel present detailed information about the entity that was selected. Hyperlinks allow easy navigation among items and edits. This provides a convenient alternative to the PDF version of the data specs.

5.2 Data Dictionary Files

- **MDS data dictionary tables (vn.nn.r).mdb** This is the Microsoft Access database that contains all of the MDS data dictionary tables that were used to generate the reports listed above. Additional reports are also available in the database.
- **itm_mstr (vn.nn.r).csv** A comma-separated value file containing data from the itm_mstr table in the data dictionary. This is the master item table that contains one record for each MDS item. This table could be useful for programmers who wish to build their own MDS 3.0 data dictionary.
- **itm_val (vn.nn.r).csv** A comma-separated value file containing data from the itm_val table in the data dictionary. This table contains one record for every response option for each MDS item. This table can also be used in a data dictionary when linked with the item master table described above. It could also be used to generate reports or screens containing the text of each item’s response options.
- **isc_mstr (vn.nn.r).csv** A comma-separated value file containing data from the isc_mstr table in the data dictionary. This table lists the ISC codes and is useful for generating reports that describe the item subset codes.

- **isc_val (vn.nn.r).csv** A comma-separated value file containing data from the isc_val table in the data dictionary. This table lists allowable combinations of the reason-for-assessment items and the ISC code that is associated with each combination. This table can serve as a useful lookup table for converting the reasons for assessment into item subset codes.
- **itm_sbst (vn.nn.r).csv** A comma-separated value file containing data from the itm_sbst table in the data dictionary. This table lists one record per assessment item and shows the item subsets for which each item is active, inactive, or state optional. This table can be useful for determining which items are active on a particular type of record.

The fields within each of these tables are described in Appendix A of this document.

5.3 Microsoft Access Reports

As noted above, one of the files that is distributed with the data specifications is the Microsoft Access database that contains the MDS 3.0 data dictionary. This database can be used to generate additional reports that are not distributed with the data specifications. The following is a brief description of these reports.

- **Public: data dictionary report.** This report contains a description of each table and field that is part of the data specs data dictionary.
- **Public: data specs report.** This is the same as the data specs report that is part of the distribution package.
- **Public: edit change.** This is the same as the edit change report that is part of the distribution package.
- **Public: ISC-RFA report.** This report lists all possible combinations of the RFA items and their associated ISCs. It is based upon the isc_mstr and isc_val tables.
- **Public: item change report.** This is the same as the item change report that is part of the distribution package.
- **Public: item list by item.** This report is a simple list of all MDS items, sorted in logical order.
- **Public: item list by type.** This report is a simple list of all MDS items, sorted by type (code, checklist, number, text, etc.).
- **Public: item subset matrix.** This report lists each MDS item along with the item subsets for which it is active, inactive, or state-optional. It is based upon the itm_sbst table.
- **Public: item-response report.** The report lists each MDS item along with its corresponding response options.
- **Public: undup edits by ID.** This is an unduplicated list of edits, sorted by edit ID number.
- **Public: undup edits by type.** This is an unduplicated list of edits, sorted by type (none-of-the-above, skip pattern, format, consistency, etc.).

6 Detailed Data Specifications Report

The Detailed Data Specifications Report contains at least one page for every item in the MDS 3.0 item set. Each item begins on a new page. The report is divided into six major sections:

1. Basic information.
2. Item subsets for which the item is active, inactive, and state-optional
3. Allowable responses or values for the item

4. Fatal and warning edits associated with the item.
5. Supplemental information about the item (this section appears only for certain items).
6. Version notes describing changes to the item and the edits that apply to it.

Each of these sections is described below.

6.1 Basic Item Information

The top section presents basic information about the item under the following headings:

- **Item.** The item identifier (e.g., B0100).
- **Description.** A brief description of the item (e.g., “comatose”).
- **Item Group.** There are four groups of items:
 - **Control items.** Control items are supplemental items that are included in the submission file and are used to control processing or for other purposes (e.g., whether the file is a production or test file, the name of the software that was used to produce the submission file, etc.).
 - **Assessment items (abbreviated “asmt” on the report).** Assessment items are items that are part of the complete MDS 3.0 Item Set. This includes:
 - Federal items which are defined by the federal government for inclusion on submitted records. Federal items are in Sections A – R and T – Z of the MDS 3.0.
 - State items which are items defined by the States and approved by CMS for inclusion on records submitted from providers in their State. State items are always included in Section S,
 - **Calculated items (abbreviated “calc” on the report).** These items are calculated by the ASAP system, stored in CMS’s national database, and will be included in fixed-format files that are produced by CMS. ***These items are not submitted and are not to be included on the XML submission files.*** The structure and use of the fixed-format file layout is described in a later section of this document.
 - **Filler items.** Filler is reserved for future use on fixed-format files containing MDS 3.0 data. ***These items are not submitted and are not to be included on the XML submission files.*** The structure and use of the fixed-format file layout is described in a later section of this document.
- **LOINC code.** In order to promote the use of electronic health records and standardized nomenclature systems, CMS will support the optional submission of LOINC codes that are associated with MDS items and with the each of the responses to those items. MDS software that supports electronic health records and standardized transmission protocols, such as HL7 (<http://www.hl7.org>), should benefit from the assignment of LOINC codes to MDS items and responses. At this time, LOINC codes have not been assigned, but when they are available relevant LOINC codes will be displayed on this report.
- **Item type.** Items are classified into the following types:
 - **Text.** Items are those that contain text (e.g., A0500C, resident last name).
 - **Code.** Coded items are those that have a limited number of response options (e.g., B0100, Comatose, has three valid response options).

- **Checklist.** Checklist items are a subset of coded items for which each component item in the checklist has response options of 0 (Not checked (No)), or 1 (Checked (Yes)). There are two types of checklists:
 - **None-of the-Above Checklists** where the component items include a “none of the above” component that is “checked” when all of the other component items are “unchecked”. An example checklist is C0900, memory/recall ability.
 - **Other Checklist** where the final item can be checked in addition to other items being checked. An example of this checklist is X0900 Reason for Modification.
- **Number.** Numeric items can contain a range of numeric values (e.g., K0200A, height, or K0200B, weight).
- **Date.** Examples of date items include A0900, birth date, and A2300, assessment reference date.
- **ICD.** The I8000 ICD items contain diagnosis codes. Currently the MDS accepts only ICD-9 codes, but in the future ICD-10 codes will be accepted. ICD items must conform to a specific format that is defined in the data specifications.
- **Max length.** This property shows the maximum number of characters or bytes that the submitted item may contain.
- **Fixed format start-end bytes.** This column displays the start and end bytes that will be used to store the item on fixed-format files. The structure and use of the fixed-format file layout is described in a later section of this document.

6.2 Item Subsets

The item subsets section contains three lines: active, inactive, and state optional. These three lines list the ISC codes that apply to the item. For example, item E0300 has the following ISCs listed:

Active: NC

Inactive: NS,NSD,NO,NOD,ND,NT,SP,SS,SSD,SO,SOD,SD,ST,XX

State optional: NQ,NP

This means that E0300 is active on NC records (nursing home comprehensive assessments) and would therefore always be included in XML files for NC assessments. It is inactive for the types of records listed (NS, NSD, etc.) and would not be included in XML files for such records. Finally, it is state optional on NQ and NP records (nursing home quarterly and PPS assessments). This means that E0300 is not normally active on such assessments, but that individual States have the option of adding it to NQ and NP assessments in which case it must be added to XML files containing such assessments in those States.

6.3 Item Values

The table in the third section of the page lists the allowable values that may be submitted for the item. For example, three values are listed for item B0100: 0, 1, and - (dash). For each value, the LOINC code and the text associated with the value are listed. As noted above, at this time LOINC codes are unavailable but will be added in a later version of the data specifications. The text associated with each item value is taken directly from the MDS item set, where available. Some values and their associated text are not listed on the item set itself (like “-” and “^”) and will be discussed in more detail in a later section.

Note that when the text for a response option contains directions for a skip pattern, that text is omitted from the item value listed. For example, on the comprehensive assessment, option “0” says “No→Continue to B0200, Hearing”. In the data specs report, the text after “no” is omitted. The reason for

this is that the skip text sometimes changes depending upon the type of assessment and the items included on that assessment.

6.4 Item Edits

The table in the fourth section of the page lists the fatal and warning edits that are associated with the item. This table contains the following four columns:

- **Edit ID.** Each edit has been assigned an edit ID code. These codes begin with the number -3501 and increase sequentially in absolute value. The order of the edit IDs is arbitrary. These edit ID codes will be used on the feedback reports that are produced by the ASAP system. This will make it possible to directly relate an error or warning on the feedback reports with a specific edit in the data specifications.

Please note that *in the Detailed Data Specifications Report, edits are listed under every item that they apply to.* Thus, a given skip pattern edit, for example, will be listed under the item that triggers the skip pattern as well as under every item that may be skipped according to the value of the trigger item. The unique edit IDs unambiguously identify these edits that apply to more than one item.

A second report described below (the Unduplicated Edit Report), lists each edit only once and references all of the items that each edit applies to. This system of uniquely and unambiguously identifying edits is intended to assist developers in insuring that all required edits are incorporated in their software.

- **Edit Type.** As noted above, there are various types of edits which are described below.
 - **Format.** Format edits specify special rules for formatting item values.
 - **Consistency.** Consistency edits define logical constraints among multiple items.
 - **Skip pattern.** Skip patterns always involve two or more items. The first item in the group is designated the trigger (or gateway) item. The value of the trigger item determines whether the remaining items are answered by the assessor or are skipped. If an item is skipped, it will be blank (have no value). Items that are blank because they are skipped must be denoted with the “caret” character (^) in the submission file. If an item is not skipped, it will have a value other than the “caret” character. In Item B0100, for example, contains skip pattern edit -3519. This edit has two subparts (designated “a”, and “b”). These edits are related to a trigger item (B0100) and to a set of dependent items (B0200 through C0100). The first part of the edit says that if B0100 is equal to zero, the dependent items must not be skipped (must not be blank). The second part of the edit says that if B0100 is equal to one, the dependent items must be skipped (must be blank). All skip patterns follow a similar pattern (in fact, the text of these edits is automatically generated from control tables in the data specifications Access database). Skip pattern edits are listed for every item involved, including both the trigger item and all dependent items.
 - **None of the above.** These edits specify rules for none of the above checklist items. As noted above, checklist items consist of a group of component items all of which have values of 0 (Not checked (No)), and 1 (Checked (Yes)). Furthermore, the checklist group always includes a “none of the above” item. Checklist items always have three none-of-the-above subparts (designated “a”, “b”, and “c”) associated with them
 - Items C0900A through C0900Z, for example, contain edit -3502 which is a none-of-the-above edit. This edit consists of the following three parts. The first subpart says that if the “none of the above” item is equal to zero (is not checked), then at least one of the active component items must have a value of one (checked). The second subpart says that if the “none of the above” item is equal to one (checked), then all of the active component items must have a value of zero (not checked). Finally, the third subpart says

that if the “none of the above” item has a value of dash (not assessed or information not available), then at least one of the active component items must equal dash and the remaining active component items must equal zero or dash. All “none of the above” edits will follow this exact same pattern (in fact, the text of these edits is automatically generated from control tables in the data specifications Access database). The same none-of-the-above edits will be listed on the report for every item in the checklist group.

- **Severity.** The severity column describes the impact of violating the edit. There are two possible values:
 - **Fatal.** Violation of a fatal edit will result in rejection of the submission file. Format, none-of-the-above, and skip pattern edits are always fatal. Most consistency edits are fatal, but some are warnings.
 - **Warning.** Violation of a warning edit will result in a warning message on the user feedback report. However, a warning will not prevent the submitted assessment data from being accepted and stored in the ASAP system.
- **Edit Text.** This column contains the text of the edit.

6.5 Supplemental Information

The fifth section of the report is a Supplemental Information table. This table appears for only a few items. It contains a list of one or more informative messages that provide background information or additional instructions that are related to the item. These messages are not issued by the ASAP system. This table contains the following columns:

- **Info ID.** This is a unique ID that is assigned to the message. Info IDs begin with the number - 9001 and increase sequentially in absolute value. The order of the edit IDs is arbitrary.
- **Type.** The type of message is always “Information” for information messages.
- **Text.** This column contains the text of the message. For example, item A0100C, State provider number, has a supplemental information entry.

6.6 Version Changes

The final section of the report lists any changes that were made to the item or the edit since the previous version of the data specs was released. This section will appear only for items where a change has been made. This section will not appear in the initial release of the data specifications, but will be included in subsequent releases.

7 Unduplicated Edit Report

As noted in the previous section, the Detailed Data Specifications Report lists all of the edits that are associated with each item in the MDS data set. Because most edits apply to multiple items, there is a great deal of duplication on this report. For this reason, a second report is provided which lists each edit only once. The Unduplicated Edit Report lists each edit as well as the items that it applies to.

For each edit listed, the edit ID, type, and text of the edit are displayed. After this, the items to which the edit applies are listed.

This report should serve as a resource for developers who wish to insure that their software incorporates all required edits and that each of those edits is applied to the proper set of items.

8 Conventions Used in the Data Specification Reports

Certain conventions have been adopted in the data specification reports in order to make them clear and unambiguous. These conventions are described below.

- On the Detailed Data Specifications report, the “Item Values” table lists all allowable values for each item. If a submission file contains any values other than those listed in this table, a fatal error will occur and the file will be rejected. For example, for item A0800 (gender) the values 0, 1, and - (dash) are listed in the “Item Values” table. If any other value is submitted for A0800, a fatal error will occur. Note that edits may constrain the list of allowable values based upon specific logical conditions (e.g., if Item A has certain values, then only a subset of Item B’s values may be allowed). However, it is never allowable to submit a value that is not listed in the “Item Values” table.
- If the item is a numeric item, then the “Item Values” table will not list every individual value (because enumerating all possible values is not practical). Instead, the first two rows of the “Item Values” table will list the minimum and maximum allowable values. Restrictions on the values between the minimum and maximum values are listed in the edits for the item. Any additional rows will list special values that may be submitted. For example, the “Item Values” table for C0500 (summary score for the Brief Interview for Mental Status) has five rows. The first row lists a value of “00” which is labeled “minimum”, the second row lists a value of “15” which is labeled “maximum”, and the remaining rows list special values (99, -, and ^). C0500 has an edit with a format restriction that requires that the numbers be integers. Decimal points are not allowed.
- Two special values have been reserved for use on certain items:
 - Dashes (-) are used to indicate that an item was not assessed or the information was not available. For example, if a resident is in a facility for only a few days, it may not be possible to complete the entire assessment. In this case, the assessor may indicate for certain items that the item was not assessed. Dashes are allowed on most, but not all, items. When a dash is allowed for an item, it will be listed in the “Item Values” table. A dash must not be submitted for items where the “Item Value” table does not list it as an allowable value; submitting dashes for such items will result in a fatal error. When a dash is allowed for not assessed, a single dash should be submitted for the item regardless of the item’s normal length. Also note that dashes have a somewhat different use for the Therapy End Date items (O0400A6, O0400B6, O0400C6) and for the date of the oldest Stage 2 pressure ulcer (M0300B3). For ongoing therapy, the end date is filled with 8 dashes. Please refer to the edits for the therapy end dates for more details.
 - Carets (^) are used in the submission file to indicate that an item has been left blank due to a skip pattern or, for certain text items, that the item has been left blank by the assessor. If an item is active for given type of assessment but has been skipped because it is in a skip pattern, then the XML tags for the item must be included in the XML file and a single caret must be submitted as the value between the element’s tags. Note that carets are allowed for only some items, as indicated by the “Item Values” table. Also note that carets have a somewhat different use for the diagnosis code items (I8000A through A8000J). Like other items, a single caret is used to indicate that a given diagnosis code is entirely blank. For non-blank diagnosis codes, carets are used to indicate blank characters within the code itself. Please refer to the edits for the diagnosis code items for more details.
- Where edits refer to values of an item, those values are always enclosed in brackets. For example, an edit might say, “If C0500=[-,99], then if C0600 is active it must equal [1,-]”. This statement means, “If the value of C0500 is equal to ‘-’ (dash) or the value of C0500 is equal to ‘99’, then if C0600 is active its value must be equal to ‘1’ (one) or ‘-’ (dash)”. The values contained within brackets should be understood to be character literals even though quotation marks have been omitted. Furthermore, when more than one value is listed, they are implicitly connected by a logical “OR”. The following summarizes the conventions that are used when specifying values:
 - [1,2,3] means “1” or “2” or “3”.

- [00-15] means “00” through “15” (inclusive).
 - [^] means the caret character, which indicates a blank.
 - “Not equal [^]” refers to any legal value for an item other than the caret character (which indicates a blank).
 - [-] refers to a dash (which indicates that an active item was not assessed or no information was available).
- The relational edits that are included in the data specifications apply only to items that are active for a particular item subset. Items that are not active on a particular item subset should not be submitted and are not edited even if they are submitted.

For example, consider an edit that says “If Item A=[1], then all active Items B, C, and D must equal [2]”. If Item A was equal to [1], then any of the items B, C, and D that were active must equal [2]. However, if any of these three items (e.g., Item B) was inactive, it would not be submitted, would not have a value, and would not be edited. The edit would therefore not apply to the inactive item but would continue to apply to the remaining active items, if any. Similarly, if Item A was not active, the entire edit would not apply.

There are exceptions to this general rule. One exception is that “none-of-the-above” edits apply only if all items involved are active. For example, one such edit is -3504 which states “If F0800A through F0800T and F0800Z are all active, then the following rules apply:...”. This none-of-the-above edit would not apply if any of the component items was inactive.

All other exceptions to the general rule are handled explicitly in individual edits.

9 XML File Structure

As noted above, MDS 3.0 data will be submitted using XML files. XML files must employ ASCII character encoding. Figure 1, below, shows how MDS XML submission files will be structured.

Figure 1: Example MDS 3.0 XML File

```

<?xml version="1.0" standalone="yes"?>
<ASSESSMENT>
  <ASMT_SYS_CD>MDS</ASMT_SYS_CD>
  <ITM_SBST_CD>NC</ITM_SBST_CD>
  <ITM_SET_VRSN_CD>1.00</ITM_SET_VRSN_CD>
  <SPEC_VRSN_CD>1.00</SPEC_VRSN_CD>
  <PRODN_TEST_CD>P</PRODN_TEST_CD>
  <STATE_CD>IA</STATE_CD>
  <FAC_ID>1231_B</FAC_ID>
  <SFTWR_VNDR_ID>12321345</SFTWR_VNDR_ID >
  <SFTWR_VNDR_NAME>SOME VENDOR</SFTWR_VNDR_NAME >
  <SFTWR_VNDR_EMAIL_ADR>SUPPORT@VENDOR.COM</SFTWR_VNDR_EMAIL_ADR>
  <SFTWR_PROD_NAME>MDS ENTRY SYSTEM</SFTWR_PROD_NAME>
  <SFTWR_PROD_VRSN_CD>V2.44</SFTWR_PROD_VRSN_CD>
  <FAC_DOC_ID>A1334001</FAC_DOC_ID>
  <A0100A>1234567890</A0100A>
  <A0100B>123456</A0100B>
  <A0100C>M33298483</A0100C>
  .
  .
  .
  <A0800>1</A0800>
  <A0900>19350621</A0900>
  .
  .
  .
  <C0800 LOINC_ITEM="99999-9">1</C0800>
  <C0900A LOINC_ITEM="99999-9" LOINC_RESP="99999-9">0</C0900A>
  <C0900B LOINC_ITEM="99999-9" LOINC_RESP="99999-9">1</C0900B>
  <C0900C LOINC_ITEM="99999-9" LOINC_RESP="99999-9">1</C0900C>
  <C0900D LOINC_ITEM="99999-9" LOINC_RESP="99999-9">1</C0900D>
  <C0900Z LOINC_ITEM="99999-9" LOINC_RESP="99999-9">0</C0900Z>
  <C1000 LOINC_RESP="99999-9">-</C1000>
  .
  .
  .
  <E0300>0</E0300>
  <E0500A>^</E0500A>
  <E0500B>^</E0500B>
  <E0500C>^</E0500C>
  <E0600A>^</E0600A>
  <E0600B>^</E0600B>
  <E0600C>^</E0600C>
  <E0800>3</E0800>
  <E0900>2</E0900>
  .
  .
  .
  <Z0500>20101001</Z0500>
</ASSESSMENT>

```

The XML file depicted in Figure 1 shows only a subset of the elements (items) that might be submitted. The dots are meant to depict additional elements that would be included in the submission file. Please note that the LOINC codes in Figure 1 are always "99999-9". These are fictitious values that are included solely for illustrative purposes and would be replaced with actual LOINC codes when they become available.

The example above begins with the standard XML header line which is followed by the <ASSESSMENT> element that contains the data for a single assessment. Only a single assessment may be included in a

submission file. If a facility wishes to submit data for multiple assessments during an upload session, separate files must be created for each assessment. These separate files should be zipped together into a single zip file which can then be uploaded.

The ASAP system will process only zip files. Any submitted file that is not a zip file will be rejected. All submission files must be 5 MB or less in size. Any submission file exceeding this size limitation will be rejected by the system. If a ZIP file contains multiple XML files, the ASAP system will sort the data within the ZIP file before processing. This allows proper processing of MDS records when multiple records are submitted for the same resident, as long as those multiple records are included in the same ZIP file. If multiple records for the same resident and target date are being submitted at the same time, these records should therefore be included in the same ZIP file. If they are spread across two or more ZIP files, unexpected timing errors may occur.

The following rules must be followed for naming XML and ZIP submission files:

1. File names for ZIP files cannot exceed 260 characters, including the file extension. A file extension of “.zip” is required.
2. File names for XML files cannot exceed 260 characters, including the file extension. A file extension of “.xml” is recommended, but is not required.

The <ASSESSMENT> beginning tag and the </ASSESSMENT> ending tag are used to enclose the elements for individual items that belong to the assessment. These tags are required. The file must include elements for every item that is active for the type of assessment that is being submitted. In addition, any State-optional items that are required in the facility's State must also be submitted. Any other elements that are included in the submission file will be ignored by the ASAP system and any data contained in those elements will not be stored in CMS's database.

The tag for each item corresponds to the item IDs that are listed in the Detailed Data Specifications Report. For example, the beginning tag for item A0100A is <A0100A> and the ending tag is </A0100A>. The submitted value for each item is included within that item's tags. For example, in Figure 1 the value of A0800 is “1”. The <ASSESSMENT> beginning tag, the </ASSESSMENT> ending tag, and all intervening tags must be upper case. These tags are required

For numeric items, leading and trailing zeroes may be omitted. For items that can contain a decimal value, the decimal point must be included if fractional amounts are included, but may be omitted if an integer value is being submitted. For example, item M0610A is used to report the length of a pressure ulcer in centimeters and includes up to one decimal value. If the value being submitted is 1.2 centimeters, then the following alternatives are acceptable:

```
<M0610A>1.2</M0610A>
<M0610A>01.2</M0610A>
```

If the value being submitted is 1.0 centimeters, then the following alternatives are acceptable:

```
<M0610A>1.0</M0610A>
<M0610A>1.</M0610A>
<M0610A>1</M0610A>
<M0610A>01.0</M0610A>
<M0610A>01.</M0610A>
<M0610A>01</M0610A>
```

For items that can contain only an integer value, no decimal point is allowed in the submitted value. For example, item C0500 is used to report the BIMS Summary Score. This is an integer item so will only accept the following integer values:

```
<C0500>0<C0500>
<C0500>1<C0500>
<C0500>2<C0500>
<C0500>3<C0500>
```

```

<C0500>4<C0500>
<C0500>5<C0500>
<C0500>6<C0500>
<C0500>7<C0500>
<C0500>9<C0500>
<C0500>00<C0500>
<C0500>01<C0500>
<C0500>02<C0500>
<C0500>03<C0500>
<C0500>04<C0500>
<C0500>05<C0500>
<C0500>06<C0500>
<C0500>07<C0500>
<C0500>09<C0500>
<C0500>10<C0500>
<C0500>11<C0500>
<C0500>12<C0500>
<C0500>13<C0500>
<C0500>14<C0500>
<C0500>15<C0500>

```

Note that these rules do not apply to the diagnosis code items (I8000A through A8000J) which are not numeric and which have their own specific formatting rules (refer to the specifications for those items for details).

Dates must be submitted in YYYYMMDD format (see item Z0500 in Figure 1 for an example). The exception is a birth date where the day or the month and day may be unknown. If the full birth date is known, it must be submitted as YYYYMMDD like any other date. For example, if the birth date is April 17, 1935, it would be submitted as:

```
<A0900>19350417</A0900>
```

If the year and month were known, but not the day, it would be submitted as:

```
<A0900>193504</A0900>
```

If only the year were known, it would be submitted as:

```
<A0900>1935</A0900>
```

Note that for all items except the diagnosis codes, leading and trailing blanks should be trimmed. In addition, alphabetic text in any item (such as resident name) may be submitted in either upper, lower, or mixed case. The ASAP system will trim leading and trailing blanks on all submitted values except the diagnosis codes (contained in I8000) and will convert alphabetic text except for the software vendor's e-mail address (SFTWR_VNDR_EMAIL_ADR) to upper case without issuing any warnings. These converted values will be used on submission feedback reports and other database reports. Thus, users should be aware that even if a text item (such as resident name) is submitted as a lower case string or with leading or trailing blanks, it will appear trimmed and in upper case in the feedback reports.

If the value of an item in the XML file exceeds the maximum length of the item, the item will not be parsed and a fatal error will be issued. Some MDS items (such as A0500C, resident last name) can contain special characters, such as apostrophes. A properly formatted XML file must encode these characters using certain conventions. For example, the name "O'NEAL" would be encoded as "O&APOS;NEAL". It is possible that such an item, in its raw, XML form before it is parsed, could violate MDS edits. For example, a string such as "O&APOS;NEAL" could be longer than the maximum allowed length for an item or might contain characters (such as the ampersand) that are not allowed for the item. Such items **will** be accepted, however, because the edits are applied **after** the XML file has been parsed. The parsing will

convert the XML coding of the special characters to the desired character. In the case of A0500C, “O&APOS;NEAL” would be parsed to “O’NEAL” before any edits are applied.

Each element may contain either of two attributes: (a) LOINC_ITEM (the LOINC code associated with the item) and (b) LOINC_RESP (the LOINC code associated with the item’s response value). These attributes are optional; both, either, or neither of these attributes may be included with each element. No editing of these attributes will occur at this time. If these attributes are included, they must be syntactically correct (according to XML rules). In no other respect will their presence affect the processing of the XML file. LOINC codes that are included in the submission files will not be checked to insure that they correspond to the LOINC codes that are assigned to the items or values. These LOINC attributes are allowed solely to allow software developers to include them in the submission file if they are needed for other purposes. The LOINC attribute tags (“LOINC_ITEM” and “LOINC_RESP”) should be upper case for consistency. Figure 1 shows examples where both LOINC attributes are included (e.g., C0900A), where one or the other are included (e.g., C0800 or C1000), and where neither are included (A0800).

Figure 1 also illustrates the use of the special characters described above: dashes and carets. Item C1000 contains a dash, indicating that the item was not assessed or the information was not available. Item E0300 contains a “0” which triggers a skip pattern whereby the items E0500A through E0600C are skipped and the assessment continues with E0800. It can be seen in Figure 1 that the skipped items, E0500A through E0600C, contain carets.

10 Item Subset Codes

10.1 Introduction

Item subset codes (ISCs) have been defined that correspond to the various types of MDS records that can be submitted. The ISCs serve a similar function as the record types (REC_TYPES) that were defined for MDS 2.0. Table 2, below, defines the ISCs that are used for MDS 3.0.

Table 2: Item Subset Codes

ISC	Description	Item Groups								
		Demo-graphic and Admin	QI items	QM items	CAA items	RUG rehab grp items	RUG non-rehab items	Survey and Certifi-cation items	State-optional items	Section X (Modifica-tion/Inac-tivation)
Nursing Home ISCs										
NC	Comprehensive	x	x	x	x	x	x	x		x
NQ	Quarterly	x	x	x		x	x	x	x	x
NP	PPS	x	x	x		x	x	x	x	x
NS	OMRA SOT	x				x				x
NSD	OMRA SOT + Discharge	x		x		x				x
NO	OMRA other	x				x	x			x
NOD	OMRA other + Discharge	x		x		x	x			x
ND	Discharge	x		x						x
NT	Tracking (entry/expired)	x								x
Swing Bed ISCs										
SP	PPS	x		x		x	x			x
SS	OMRA SOT	x				x				x
SSD	OMRA SOT + Discharge	x		x		x				x
SO	OMRA other	x				x	x			x
SOD	OMRA other + Discharge	x		x		x	x			x
SD	Discharge	x		x						x
ST	Tracking (entry/expired)	x								x
Inactivations										
XX	Inactivation	x								x

Note: OMRA=other Medicare required assessment, SOT=start of therapy.

The top portion of the table describes nursing home ISCs (which begin with “N”) and swing bed ISCs (which begin with “S”). The last row represents inactivations¹, which are identical for nursing home and swing bed assessments. The columns in Table 2 represent different groups of MDS items such as items used for quality indicators (QIs), quality measures (QMs), Care Area Assessment (CAA), etc². Cells marked with “x” in the body of the table show the item groups that are represented on each ISC. For example, a nursing home quarterly assessment (NQ) has demographic and administrative items, QI and QM items, both types of RUG items, items needed for survey and certification purposes, and Section X items. It also allows for federally defined comprehensive items that States may choose to include.

It can be seen that the items that are included on a particular type of record is driven by the item groups that are associated with the ISC. The following two sections will describe how one determines the ISC that is associated with an MDS record and how one then determines which items are associated with that ISC.

¹For a description of inactivation procedures, please refer to Chapter 5 of the RAI manual.

²Note that there is considerable overlap among the items that are included in each of the item groups listed (i.e., the item groups are not mutually exclusive)

10.2 Determining the ISC for an MDS Record

The ISC is determined from the reasons for assessment for the MDS record. The RFA items are as follows:

- A0200: type of provider (nursing home vs. swing bed)
- A0310A: OBRA reason for assessment
- A0310B: PPS reason for assessment
- A0310C: PPS other reason for assessment (OMRA)
- A0310D: swing bed clinical change assessment
- A0310F: entry/discharge reporting

There are 4,200 combinations of the possible values of these six RFA items. Most of these (3,346) are combinations of values that are not allowed (i.e., that will lead to record rejection). The remaining combinations can be mapped onto the ISC codes described above. Unfortunately, the logic for determining the ISC is not straightforward and cannot be reduced to a set of simple rules. To assist programmers, we have provided two options for determining the ISC code from the RFA items.

The first option is to use a lookup table that is supplied with the data specifications. In the Access database, this table is called `isc_val`. The contents of this table are supplied with the data specs in a comma separated value file called `isc_val.csv`. This table contains one record for each of the 4,200 combinations of the ISC items. It also includes one additional record that corresponds to an inactivation ISC ("XX"). Each record contains a unique combination of the RFA items in the fields named "_val" (i.e., in `A0200_val`, `A0310A_val`, etc.). The ISC that is associated with the RFA combination is in the field called `isc_id`. If this field contains dashes ("--"), the combination of RFA values is not allowed.

Instead of using the lookup table, the programmer can implement the logic that is shown in Appendix B. This appendix contains the source code for a Visual Basic function that accepts the values of the six RFA items as string input, and returns the ISC value as a string. If the RFA combination is invalid, the function will return dashes ("--").

Note that early versions of the `isc_val` table contained a special row associated with the "XX" ISC. Beginning with Version 1.00.2 of the data specs, that row is no longer contained in the table. Because this row was removed, the lookup table should be used only for records that are not inactivation records (where `A0050` (formerly `X0100`)=[1,2]). The ISC is "XX" when `A0050`=[3], indicating an inactivation.

10.3 Determining the MDS Items Associated with an ISC

Once the ISC has been ascertained, the items that are active, inactive, and state optional for the MDS record can be determined. The Access database includes a table called `itm_sbst` that contains the necessary information. The contents of this table are supplied with the data specs in a comma separated value file called `itm_sbst.csv`.

This table contains one record for each MDS submission item, and the columns correspond to the various ISCs. Each item/ISC combination in this table can have one of three values:

- x = the item is active on the ISC
- s = the item is state-optional on the ISC
- blank = the item is inactive on the ISC

Items that are active must be included in the XML submission file. Items that are inactive should be omitted from the XML file; if they are included they will be ignored by the ASAP system. Finally, items that are state optional must be included in the XML file in States that have required them (different States may opt to include all, some, or none of these optional items). Developers should note that the ISC assignments contained in the `itm_sbst` table is the definitive list that should be used for software

development. A simplified version of this table may be included with the RAI manual, but may not contain all of the information required by developers.

11 Fixed-Format File Layout

11.1 Uses for the Fixed-Format Layout

As noted above, nursing home and swing bed providers will use XML files to submit data to CMS. However, the data specs also define a fixed-format file layout which will be used in other circumstances. For example, CMS will use the fixed file format for data extracts, such as those that will be used to periodically extract data from the national database which is sent to individual States. The format will also be used to pass data to utilities (e.g., RUG groupers) that are provided by CMS. Finally, the format will also be used to produce data extracts for other users (such as researchers or individual providers who need to rebuild their assessment database), and it will also be used for test data that will be supplied with CMS-supplied software (such as RUGs DLLs). Basically, this fixed format will be useful for anyone who wishes to transfer large batches of assessment data, and software vendors may find it useful to support this format for importing MDS 3.0 data. Software which uses CMS-supplied DLLs will have to be able to build a fixed-format string of MDS 3.0 data in order to call those DLLs.

The data specifications provide information about starting and ending bytes for each item in the fixed format record. This information is also contained in the itm_mstr table in the Access database or in the itm_mstr.csv file that is supplied with the data specs. Each item's starting byte, ending byte, and length are contained in the following fields: fixed_rec_strt_byte, fixed_rec_end_byte, and fixed_rec_lngth. The table must be sorted by the field called itm_srt_id to put items in the order they will appear in the fixed format record.

Note that the table contains items that are not included in XML submission files. The field called itm_grp_cd identifies the item group: "control", "asmt" (assessment), "filler", and "calc" (calculated). Only control and assessment items are to be included in XML submission files. When a record is accepted by the ASAP system, certain calculated values are stored in CMS's national database. These calculated values will be contained in the "calc" items at the end of the fixed-format string. The filler items provide space that will be used for Federal and State (Section S) assessment items that might be added in the future.

11.2 Rules for Creating the Fixed-Format String

In order to enforce standardization, we have developed the rules below that describe how the fixed-format string must be formatted.

1. The string must be 3,690 bytes in length.
2. The last three bytes of the string must contain the following characters:
 - a. Byte 3,688 must contain the percent sign ("%") to indicate the end of data.
 - b. Byte 3,689 must contain a carriage return character (ASCII 013).
 - c. Byte 3,690 must contain a line feed character (ASCII 010).
3. Except for the three items listed above, all items that are defined as calculated items (that belong to the item group called "Calc") may be left blank. These items are contained in bytes 2,927 through 3,687. These calculated items will be populated in export files that are created by CMS for various purposes.
4. Any items belonging to the item group called "Filler" should be left blank. Data for each item must be contained within the start and end bytes defined in the data specifications.
5. MDS items that are inactive on a particular record should be filled with blanks. Any data contained in the fields for inactive items will be ignored.

6. MDS items that are active on assessment record but are blank due to a skip pattern must contain a single caret (“^”). If the length of the item is greater than 1 byte, then the single caret must be left justified and the remaining bytes in the field must be filled with blanks. For example, if a resident is comatose (item B0100=1), then many items are skipped, including B0200 and D0300. If these items were **active** for a given assessment, then B0200, a one-byte item located in byte 512 of the fixed-format record, would contain “^”. Item D0300, a two-byte item located in bytes 562-563, would contain “^ ” (a caret followed by a space). On the other hand, if both of these items were **inactive** on a given record, then both would be blank filled.
7. Many MDS items may be coded with a single dash (“-”) if the item was not assessed or information was not available to the assessor. If the length of the item is greater than 1 byte, then the single dash must be left justified and the remaining bytes in the field must be filled with blanks. For example, if item D0300, a two-byte item, which is 2 bytes in length, was coded with a dash, then “- ” (a dash followed by a blank) would be inserted in bytes 562-563 in the fixed-format record.
8. The rules below define formatting rules that are specific to each of the different data types.
 - a. **Checklist items.** All of these items are one byte in length. The value that is contained in the record (i.e., [0,1,-,^]) must be inserted in the correct byte of the fixed-format record. No special formatting is required.
 - b. **Code items.** The value inserted in the fixed-format record for a coded item must match exactly one of the values allowed in the data specifications for that item. For example, item A0310A is a two-byte coded item that allows the following values: [00,01,02,03,04,05,06,99]. The value inserted in bytes 270-271 of the fixed-format record must match exactly one of the eight values listed. For example, it is not acceptable to insert “ 1” (blank followed by a “1”) or “1 ” (“1” followed by a blank) for a value of “01”. For a few items, the allowed values that are listed are shorter than the length of the item. For example, the data specs version code, SPEC_VRSN_CD, lists an allowed value of “1.00” even though the item is 10 bytes in length. In these cases, left-justify and blank fill the value (i.e., put “1.00” followed by six blanks in bytes 24-33 of the fixed-format record).
 - c. **Date items.** All date items are eight bytes in length and are coded as YYYYMMDD. These date values must be inserted in the fixed-format string exactly as coded. For example, if a date item contained “20101108” (11/08/2010), then “20101108” must be inserted in the appropriate bytes in the fixed-format string. There are several exceptional cases:
 - i. Item A0900 (birth date) can have a missing day (in which case it is coded YYYYMM), or a missing month and day (in which case it is coded YYYY). In these cases, left-justify the coded value and fill the remainder of the field with blanks. For example, if the date of birth was coded as “1920” (i.e., the month and year were unknown), then bytes 351-358 of the fixed-format record must contain “1920 ” (“1920” followed by 4 blanks). Note that any fixed-format file that is created by CMS, the birth date will not contain a partial date because the month and/or day will be imputed where necessary.
 - ii. Several date items can be dash filled. For example, item A2400C (end date of most recent Medicare stay) can be dash filled if the Medicare stay is ongoing. Dash-filling these items should not be confused with entering a single dash when the item was not assessed or the information could not be obtained. When a date item is dash filled, all eight dashes must be inserted in the fixed-format record. For example, if A2400C was coded as dash-filled, then “-----” (eight dashes) would be inserted in bytes 503-510 of the fixed format record.
 - d. **ICD items.** The ICD diagnosis code items I8000A through I8000J have specific coding requirements that are described in detail in the data specs. These coding requirements do not allow for left- or right-trimming of the items. Characters of the ICD code must be in specific positions within the item and carets (which stand for blanks) are an integral part of the coded item. These items must be inserted in the fixed-format record exactly as coded and in conformance with the rules described in the data specs. For example, if item I8000A

contained the value “^^123.4^”, then that exact value would be inserted in bytes 751-758 of the fixed-format string.

- e. **Number items.** The value inserted in the fixed-format string for a numeric item must match exactly one of the values (or the range of values) allowed in the data specifications for that item. This means that numeric values must be right-justified and zero-filled. For example, item D0300 can have the following values: [00-27,99,-,^]. If the value for an MDS record is “01”, then “01” must be inserted in bytes 562-563 of the fixed-format record; “1 ” (one followed by a blank) or “ 1” (a blank followed by one) are not allowed. As with other items, however, special codes (dash and caret) must be left-justified and blank filled. Therefore, if the value for an MDS record is “-”, then “- ” (dash followed by a blank) must be inserted in the fixed format record.

The fixed-format numeric values for items that contain decimals (e.g., item M0610A) must also match exactly the values listed in the data specs. For example, if M0610A had a value of “01.0”, then “01.0” must be inserted in bytes 920-923 of the fixed format record. Alternative representations which omit zeroes or the decimal are not allowed.

Note that these rules differ from the rules that apply to XML submission files. For example, if the value of D0300 on an MDS record is “01”, a value of either “01” or “1” may be submitted in an XML file. However, for the fixed-format record, a value of “01” must be used. Similarly, if the value of M0610A is equal to “01.0”, then values such as “1”, “1.”, “1.0” may be submitted in the XML file. However, the value “01.0” must be used in the fixed-format record.

- f. **Text items.** Text items (such as A0500C, resident last name) can have a large set of possible values and the data specifications therefore cannot delineate all allowed values. Furthermore, the values for these items can be shorter than the maximum allowed length. Text values must therefore be left-justified and blank filled in the fixed-format record. For example, if a resident’s last name is “Smith”, then “SMITH ” (“SMITH” followed by 13 spaces) must be inserted in bytes 293-310 of the fixed-format record. For consistency, all text items (except SFTWR_VNDR_EMAIL_ADR, software vendor email address) should be converted to upper case before inserting them in the fixed-format record, although this is not required. It is acceptable to use lower case characters for SFTWR_VNDR_EMAIL_ADR, since email addresses are typically lower case.

12 Section S

On MDS 2.0, a portion of the submission record was set aside for state-defined assessment items. This section was referred to as “Section S”. MDS 3.0 also supports Section S. The items that CMS has approved for use in Section S were added to the data submission specifications starting with Version 1.00.2.

For MDS 3.0, there is a single set of Section S items that can be used by any State. With CMS approval, states will be allowed to collect all, some, or none of the Section S items on approved ISC’s. As with MDS 2.0, Section S items will be edited for valid ranges and formats, but there will be no consistency edits or other inter-item edits applied. For errors in valid values or formats on Section S items, a warning message will be issued. These errors will not cause the assessment to be rejected. Section S items with format or valid value errors will not be stored in the QIES database and will not be available to the states.

In the future, States will be allowed to request that CMS add new items to the list of available Section S items. If the new state requested item is approved, CMS will add the item to the approved list on a periodic basis. When this happens, a new version of the data specs will be produced. The data specs will *not* contain information about the Section S items that have been selected by each State. As with MDS 2.0, software vendors must obtain that information from individual States.

13 Additional Documentation

In order to understand the submission process completely, software developers will need information that is not contained within the data specs themselves or in the current document. This additional information

is available in the RAI User's Manual that is being published by CMS. The RAI manual contains information about topics such as submission timing, record sequencing rules, and record modification and inactivation procedures. Detailed specifications for RUGs, CAAs, QMs, and QIs are being published in separate documents that may also be of use to software developers.

Appendix A: Data Dictionary Files

As noted above, the data dictionary that was used to produce the data specifications are distributed to assist software developers. The first of these files is the Microsoft Access database (MDB file) that was used to store the data dictionary tables. In addition, the data dictionary tables are distributed as a set of comma-separated value (CSV) files. The most useful tables that are contained in the database are described below.

Table A1: Database Table Descriptions

Table Name	Description
itm_mstr	Master table containing one record for every item that is contained in the MDS 3.0 item set.
itm_val	Detail table that contains one record for every value (response option) that is allowed for each item. This table is linked to the itm_mstr table using the itm_mstr_key field.
rltn_txt	Contains one record for every edit or information message. The text of each message is stored in each record.
rltn_itm_txt	Contains one record for every edit or information message that is associated with every item. This table was used to generate the detailed data specifications report, the unduplicated edits report, and the supplemental information report.
isc_mstr	Master table containing one record for every item subset code (ISC).
isc_val	Detail table that lists the values of the reason for assessment items that are associated with each item subset code (ISC). This table is linked to the isc_mstr table using the isc_mstr_key field.
itm_sbst	Contains one record per MDS item and one column per ISC. Indicates whether each item is active, inactive, or state-optional on each ISC. An "x" indicates the item is active, "s" indicates state-optional, and a blank indicates the item is inactive on the ISC. This table is linked to the itm_mstr table through the itm_mstr_key field.

The following table describes the fields that are contained in each of the database tables described above.

Table A2: Database Field Descriptions

Table	Field	Data Type	Field Size	Description
isc_mstr	isc_mstr_key	Number	4	primary key
isc_mstr	isc_id	Text	3	ISC code
isc_mstr	isc_txt	Text	55	ISC description
isc_val	isc_id_key	Number	4	primary key
isc_val	isc_id	Text	10	ISC ID code
isc_val	isc_mstr_key	Number	4	foreign key
isc_val	A0200_val	Text	255	A0200 value
isc_val	A0200_txt	Text	12	A0200 value text
isc_val	A0310A_val	Text	10	A0310A value
isc_val	A0310A_txt	Text	25	A0310A value text
isc_val	A0310B_val	Text	10	A0310B value
isc_val	A0310B_txt	Text	25	A0310B value text
isc_val	A0310C_val	Text	10	A0310C value

Table	Field	Data Type	Field Size	Description
isc_val	A0310C_txt	Text	25	A0310C value text
isc_val	A0310D_val	Text	10	A0310D value
isc_val	A0310D_txt	Text	25	A0310D value text
isc_val	A0310F_val	Text	10	A0310F value
isc_val	A0310F_txt	Text	25	A0310F value text
itm_mstr	itm_mstr_key	Number	4	primary key
itm_mstr	sys_cd	Text	10	"MDS", "OASIS", "IRF-PAI", "CARE"
itm_mstr	form_vrsn	Text	10	form version (e.g., "1.00")
itm_mstr	spec_vrsn	Text	20	data specs version (e.g., "1.00")
itm_mstr	itm_srt_id	Number	4	item sort sequence (e.g., 12600)
itm_mstr	itm_id	Text	30	item ID code (e.g., "C0100")
itm_mstr	itm_db_id	Text	30	item database ID (e.g., "C0100_HEARG")
itm_mstr	itm_shrt_label	Text	50	item short label (e.g., "Hearing")
itm_mstr	itm_sect_srt_id	Text	2	item section sort ID (e.g., "01", "02")
itm_mstr	itm_sect_label	Text	10	item section label (e.g., "A")
itm_mstr	itm_grp_cd	Text	10	"Asmt", "Control", "State" (section S)
itm_mstr	itm_loinc_id	Text	20	LOINC item code
itm_mstr	itm_type_cd	Text	10	"Text", "Date", "Code", "Number", "ICD9"
itm_mstr	fixed_rec_srt_id	Number	4	Sort sequence for fixed-format items (e.g., 12600)
itm_mstr	fixed_rec_strt_byte	Number	4	Starting byte for fixed format record (e.g., export record)
itm_mstr	fixed_rec_end_byte	Number	4	Ending byte for fixed format record (e.g., export record)
itm_mstr	fixed_strt_end_bytes	Text	10	String showing start and end bytes
itm_mstr	fixed_rec_lngth	Number	4	Field length for fixed format record (e.g., export record)
itm_mstr	itm_vrsn_notes	Memo	0	Notes describing changes since previous specs version
itm_mstr	itm_mnl_txt	Memo	0	Text extracted from assessment user's manual
itm_mstr	isc_active	Text	80	ISC list: item is active
itm_mstr	isc_inactive	Text	80	ISC list: item not active
itm_mstr	isc_state_opt	Text	80	ISC list: item state optional
itm_sbst	itm_sbst_id	Number	4	primary key
itm_sbst	itm_mstr_key	Number	4	foreign key
itm_sbst	itm_id	Text	255	item ID code (e.g., "C0100")
itm_sbst	itm_srt_id	Number	4	item sort sequence (e.g., 121.50)
itm_sbst	itm_grp_cd	Text	10	"Asmt", "Control", "Filler", "Calc"
itm_sbst	NC	Text	1	Nursing home: comprehensive
itm_sbst	NQ	Text	1	Nursing home: quarterly
itm_sbst	NP	Text	1	Nursing home: PPS
itm_sbst	NS	Text	1	Nursing home: start of therapy OMRA
itm_sbst	NSD	Text	1	Nursing home: start of therapy OMRA + discharge
itm_sbst	NO	Text	1	Nursing home: other OMRA
itm_sbst	NOD	Text	1	Nursing home: other OMRA + discharge
itm_sbst	ND	Text	1	Nursing home: discharge
itm_sbst	NT	Text	1	Nursing home: tracking (entry/expired)
itm_sbst	SP	Text	1	Swing bed: PPS

Table	Field	Data Type	Field Size	Description
itm_sbst	SS	Text	1	Swing bed: start of therapy OMRA
itm_sbst	SSD	Text	1	Swing bed: start of therapy OMRA + discharge
itm_sbst	SO	Text	1	Swing bed: other OMRA
itm_sbst	SOD	Text	1	Swing bed: other OMRA + discharge
itm_sbst	SD	Text	1	Swing bed: discharge
itm_sbst	ST	Text	1	Swing bed: tracking (entry/expired)
itm_sbst	XX	Text	1	Inactivation
itm_val	itm_val_key	Number	4	primary key
itm_val	itm_mstr_key	Number	4	foreign key
itm_val	val_srt_id	Number	4	value sort order within item
itm_val	itm_id	Text	30	item ID code (e.g., "C0100")
itm_val	val_id	Text	20	item value (e.g., "2")
itm_val	val_txt	Text	255	text associated with value (e.g., "Female")
itm_val	val_loinc_id	Text	20	LOINC value code

Appendix B: ISC Determination Logic

The following logic can be used to determine the ISC from the six reasons for assessment items. This function is written in Visual Basic, but can be modified for different computer languages.

It is important to note that this logic does not apply to inactivations. An inactivation record is identified by item X0100=[3]. These records will have an ISC of "XX" and all items except Section X will be inactive.

Note that the logic in this function has been updated several times. Please refer to the "version history" section of the comments towards the beginning of the source code below.

GetISC Function

```
Public Function GetIsc(A0200 As String, _
    A0310A As String, _
    A0310B As String, _
    A0310C As String, _
    A0310D As String, _
    A0310F As String) As String

    ' This function calculates an ISC code based upon the following MDS items:
    '
    '     A0200 - Type of provider
    '     A0310A - Federal OBRA reason for assessment
    '     A0310B - PPS reason for assessment
    '     A0310C - PPS other Medicare required assessment (OMRA)
    '     A0310D - Swing Bed clinical change assessment
    '     A0310F - Entry/discharge reporting
    '
    ' The function returns a string containing a two- or three-character ISC
    ' code or two dashes ("--") if the combination of input values is invalid.
    '
    ' Note that the function assumes that the five input MDS items have already
    ' been validated and have the proper ranges.
    '
    ' This code is written in Visual Basic. If you are converting this code
    ' to another language, please note the following:
    '
    ' 1. The underscore is Visual Basic's line continuation character.
    ' 2. The source code calls Visual Basic's InStr() function. As used in this
    ' program, this function is called as follows:
    '
    '     InStr(string1, string2)
    '
    ' where:
    '
    '     string1 = the string to be searched
    '     string2 = string expression being sought
    '
    ' InStr returns an integer representing the byte in <string1> where
    ' <string2> begins. If <string2> is not contained in <string1>,
    ' then InStr returns a value of zero.
    '
    ' For example:
    '     InStr("01,02,03", "02") returns a value of 4.
    '     InStr("01,02,03", "99") returns a value of 0.
    '
    '-----
    'REVISION HISTORY
    ' 10/30/2009: Initial release
    ' 03/25/2010: Revision to fix OMRA-related problem.
    '     In initial version, this function returned a value of "NC" or "NQ"
    '     if the reason for assessment items met ALL of the following conditions:
    '         A0200=1 (nursing home assessment)
    '         A0310F=10,11,99 (discharge return anticipated, OR
    '                             discharge return not anticipated, OR
```

```

'           not entry/discharge)
'           A0310C=1,2,3           (SOT OMRA or EOT OMRA or SOT/EOT OMRA)
'           A0310A=01,02,03,04,05,06 (admission OR
'           quarterly OR
'           annual OR
'           significant change in status OR
'           significant correction of prior comp OR
'           significant correction of prior quart)
'           A0310B=99           (not PPS assessment)
'           These combinations are in fact invalid. A SOT, EOT, or SOT/EOT OMRA
'           must be combined with a value of A0310B that is not equal to 99 (i.e.,
'           an OMRA must be combined with a PPS assessment).
'           There were 54 reason for assessment combinations that were assigned
'           and ISC value of NC or NQ when they should have been assigned "---"
'           (invalid combination).
'           This revision fixes this problem. The changes in the code below
'           are marked with a comment that begins with "03/25/2010".
'           05/25/2010: Revision to fix admission combined with 30-, 60-, 90-day PPS.
'           Previous versions allowed admission assessment (A0310A=01) to be
'           combined with 30-, 60-, and 90-day PPS assessments (A0310B=03,04,05).
'           These combinations are, in fact, invalid.
'           There were 36 reason for assessment combinations that were assigned
'           an ISC value of NC when they should have been assigned "---" (invalid
'           combination).
'           This revision fixes this problem. The changes in the code below
'           are marked with a comment that begins with "05/25/2010".
'           04/29/2011: Revision to handle COT assessments. In previous versions,
'           item A0310C could have the values [0,1,2,3]. Beginning with V1.02.0
'           of the data submission specs, a new value was added: "4" for "change
'           of therapy" assessments. The logic below was modified to handle
'           A0310C=[4].
'-----

```

Dim ISC As String

```

'NURSING HOME ISCs
'Note that A0310D is always skipped [^] for nursing home
'assessments.
If A0200 = "1" And _
    InStr("99", A0310A) > 0 And _
    InStr("99", A0310B) > 0 And _
    InStr("0", A0310C) > 0 And _
    InStr("^", A0310D) > 0 And _
    InStr("01,12", A0310F) > 0 Then
    ISC = "NT"
'05/25/2010: Added new ElseIf where A0310A=01.
ElseIf A0200 = "1" And _
    InStr("01", A0310A) > 0 And _
    InStr("01,02,06,99", A0310B) > 0 And _
    InStr("0", A0310C) > 0 And _
    InStr("^", A0310D) > 0 And _
    InStr("10,11", A0310F) > 0 Then
    ISC = "NC"
'05/25/2010: Removed "01" from A0310A comparison below.
ElseIf A0200 = "1" And _
    InStr("03", A0310A) > 0 And _
    InStr("01,02,03,04,05,06,99", A0310B) > 0 And _
    InStr("0", A0310C) > 0 And _
    InStr("^", A0310D) > 0 And _
    InStr("10,11", A0310F) > 0 Then
    ISC = "NC"
ElseIf A0200 = "1" And _
    InStr("04,05", A0310A) > 0 And _
    InStr("01,02,03,04,05,06,07,99", A0310B) > 0 And _
    InStr("0", A0310C) > 0 And _
    InStr("^", A0310D) > 0 And _
    InStr("10,11", A0310F) > 0 Then
    ISC = "NC"
ElseIf A0200 = "1" And _
    InStr("02,06", A0310A) > 0 And _
    InStr("01,02,03,04,05,06,99", A0310B) > 0 And _

```

```

    InStr("0", A0310C) > 0 And _
    InStr("^", A0310D) > 0 And _
    InStr("10,11", A0310F) > 0 Then
    ISC = "NQ"
ElseIf A0200 = "1" And _
    InStr("99", A0310A) > 0 And _
    InStr("01,02,03,04,05,06", A0310B) > 0 And _
    InStr("0", A0310C) > 0 And _
    InStr("^", A0310D) > 0 And _
    InStr("10,11", A0310F) > 0 Then
    ISC = "NP"
ElseIf A0200 = "1" And _
    InStr("99", A0310A) > 0 And _
    InStr("99", A0310B) > 0 And _
    InStr("0", A0310C) > 0 And _
    InStr("^", A0310D) > 0 And _
    InStr("10,11", A0310F) > 0 Then
    ISC = "ND"

'05/25/2010: Added new ElseIf where A0310A=01.
ElseIf A0200 = "1" And _
    InStr("01", A0310A) > 0 And _
    InStr("01,02,06,07", A0310B) > 0 And _
    InStr("1", A0310C) > 0 And _
    InStr("^", A0310D) > 0 And _
    InStr("10,11", A0310F) > 0 Then
    ISC = "NC"
'03/25/2010: Removed "99" from A0310B comparison below.
'05/25/2010: Removed "01" from A0310A comparison below.
ElseIf A0200 = "1" And _
    InStr("03,04,05", A0310A) > 0 And _
    InStr("01,02,03,04,05,06,07", A0310B) > 0 And _
    InStr("1", A0310C) > 0 And _
    InStr("^", A0310D) > 0 And _
    InStr("10,11", A0310F) > 0 Then
    ISC = "NC"
'03/25/2010: Removed "99" from A0310B comparison below.
ElseIf A0200 = "1" And _
    InStr("02,06", A0310A) > 0 And _
    InStr("01,02,03,04,05,06,07", A0310B) > 0 And _
    InStr("1", A0310C) > 0 And _
    InStr("^", A0310D) > 0 And _
    InStr("10,11", A0310F) > 0 Then
    ISC = "NQ"
ElseIf A0200 = "1" And _
    InStr("99", A0310A) > 0 And _
    InStr("01,02,03,04,05,06", A0310B) > 0 And _
    InStr("1", A0310C) > 0 And _
    InStr("^", A0310D) > 0 And _
    InStr("10,11", A0310F) > 0 Then
    ISC = "NP"
ElseIf A0200 = "1" And _
    InStr("99", A0310A) > 0 And _
    InStr("07", A0310B) > 0 And _
    InStr("1", A0310C) > 0 And _
    InStr("^", A0310D) > 0 And _
    InStr("10,11", A0310F) > 0 Then
    ISC = "NSD"

'05/25/2010: Added new ElseIf where A0310A=01.
ElseIf A0200 = "1" And _
    InStr("01", A0310A) > 0 And _
    InStr("01,02,06,07", A0310B) > 0 And _
    InStr("2,3", A0310C) > 0 And _
    InStr("^", A0310D) > 0 And _
    InStr("10,11", A0310F) > 0 Then
    ISC = "NC"
'03/25/2010: Removed "99" from A0310B comparison below.
'05/25/2010: Removed "01" from A0310A comparison below.
ElseIf A0200 = "1" And _
    InStr("03,04,05", A0310A) > 0 And _

```

```

    InStr("01,02,03,04,05,06,07", A0310B) > 0 And _
    InStr("2,3", A0310C) > 0 And _
    InStr("^", A0310D) > 0 And _
    InStr("10,11", A0310F) > 0 Then
    ISC = "NC"
'03/25/2010: Removed "99" from A0310B comparison below.
ElseIf A0200 = "1" And _
    InStr("02,06", A0310A) > 0 And _
    InStr("01,02,03,04,05,06,07", A0310B) > 0 And _
    InStr("2,3", A0310C) > 0 And _
    InStr("^", A0310D) > 0 And _
    InStr("10,11", A0310F) > 0 Then
    ISC = "NQ"
ElseIf A0200 = "1" And _
    InStr("99", A0310A) > 0 And _
    InStr("01,02,03,04,05,06", A0310B) > 0 And _
    InStr("2,3", A0310C) > 0 And _
    InStr("^", A0310D) > 0 And _
    InStr("10,11", A0310F) > 0 Then
    ISC = "NP"
ElseIf A0200 = "1" And _
    InStr("99", A0310A) > 0 And _
    InStr("07", A0310B) > 0 And _
    InStr("2,3", A0310C) > 0 And _
    InStr("^", A0310D) > 0 And _
    InStr("10,11", A0310F) > 0 Then
    ISC = "NOD"

' 04/29/2011: Added logic to handle A0310C=4
ElseIf A0200 = "1" And _
    InStr("01", A0310A) > 0 And _
    InStr("02,07", A0310B) > 0 And _
    InStr("4", A0310C) > 0 And _
    InStr("^", A0310D) > 0 And _
    InStr("10,11", A0310F) > 0 Then
    ISC = "NC"
ElseIf A0200 = "1" And _
    InStr("03,04,05", A0310A) > 0 And _
    InStr("02,03,04,05,07", A0310B) > 0 And _
    InStr("4", A0310C) > 0 And _
    InStr("^", A0310D) > 0 And _
    InStr("10,11", A0310F) > 0 Then
    ISC = "NC"
ElseIf A0200 = "1" And _
    InStr("02,06", A0310A) > 0 And _
    InStr("02,03,04,05,07", A0310B) > 0 And _
    InStr("4", A0310C) > 0 And _
    InStr("^", A0310D) > 0 And _
    InStr("10,11", A0310F) > 0 Then
    ISC = "NQ"
ElseIf A0200 = "1" And _
    InStr("99", A0310A) > 0 And _
    InStr("02,03,04,05", A0310B) > 0 And _
    InStr("4", A0310C) > 0 And _
    InStr("^", A0310D) > 0 And _
    InStr("10,11", A0310F) > 0 Then
    ISC = "NP"
ElseIf A0200 = "1" And _
    InStr("99", A0310A) > 0 And _
    InStr("07", A0310B) > 0 And _
    InStr("4", A0310C) > 0 And _
    InStr("^", A0310D) > 0 And _
    InStr("10,11", A0310F) > 0 Then
    ISC = "NOD"

'05/25/2010: Added new ElseIf where A0310A=01.
ElseIf A0200 = "1" And _
    InStr("01", A0310A) > 0 And _
    InStr("01,02,06,99", A0310B) > 0 And _
    InStr("0", A0310C) > 0 And _
    InStr("^", A0310D) > 0 And _

```

```

    InStr("99", A0310F) > 0 Then
    ISC = "NC"
'05/25/2010: Removed "01" from A0310A comparison below.
ElseIf A0200 = "1" And _
    InStr("03", A0310A) > 0 And _
    InStr("01,02,03,04,05,06,99", A0310B) > 0 And _
    InStr("0", A0310C) > 0 And _
    InStr("^", A0310D) > 0 And _
    InStr("99", A0310F) > 0 Then
    ISC = "NC"
ElseIf A0200 = "1" And _
    InStr("04,05", A0310A) > 0 And _
    InStr("01,02,03,04,05,06,07,99", A0310B) > 0 And _
    InStr("0", A0310C) > 0 And _
    InStr("^", A0310D) > 0 And _
    InStr("99", A0310F) > 0 Then
    ISC = "NC"
ElseIf A0200 = "1" And _
    InStr("02,06", A0310A) > 0 And _
    InStr("01,02,03,04,05,06,99", A0310B) > 0 And _
    InStr("0", A0310C) > 0 And _
    InStr("^", A0310D) > 0 And _
    InStr("99", A0310F) > 0 Then
    ISC = "NQ"
ElseIf A0200 = "1" And _
    InStr("99", A0310A) > 0 And _
    InStr("01,02,03,04,05,06", A0310B) > 0 And _
    InStr("0", A0310C) > 0 And _
    InStr("^", A0310D) > 0 And _
    InStr("99", A0310F) > 0 Then
    ISC = "NP"

'05/25/2010: Added new ElseIf where A0310A=01.
ElseIf A0200 = "1" And _
    InStr("01", A0310A) > 0 And _
    InStr("01,02,06,07", A0310B) > 0 And _
    InStr("1", A0310C) > 0 And _
    InStr("^", A0310D) > 0 And _
    InStr("99", A0310F) > 0 Then
    ISC = "NC"

'03/25/2010: Removed "99" from A0310B comparison below.
'05/25/2010: Removed "01" from A0310A comparison below.
ElseIf A0200 = "1" And _
    InStr("03,04,05", A0310A) > 0 And _
    InStr("01,02,03,04,05,06,07", A0310B) > 0 And _
    InStr("1", A0310C) > 0 And _
    InStr("^", A0310D) > 0 And _
    InStr("99", A0310F) > 0 Then
    ISC = "NC"

'03/25/2010: Removed "99" from A0310B comparison below.
ElseIf A0200 = "1" And _
    InStr("02,06", A0310A) > 0 And _
    InStr("01,02,03,04,05,06,07", A0310B) > 0 _
    And InStr("1", A0310C) > 0 And _
    InStr("^", A0310D) > 0 And _
    InStr("99", A0310F) > 0 Then
    ISC = "NQ"
ElseIf A0200 = "1" And _
    InStr("99", A0310A) > 0 And _
    InStr("01,02,03,04,05,06", A0310B) > 0 And _
    InStr("1", A0310C) > 0 And _
    InStr("^", A0310D) > 0 And _
    InStr("99", A0310F) > 0 Then
    ISC = "NP"
ElseIf A0200 = "1" And _
    InStr("99", A0310A) > 0 And _
    InStr("07", A0310B) > 0 And _
    InStr("1", A0310C) > 0 And _
    InStr("^", A0310D) > 0 And _
    InStr("99", A0310F) > 0 Then
    ISC = "NS"

```

```

'05/25/2010: Added new ElseIf where A0310A=01.
ElseIf A0200 = "1" And _
  InStr("01", A0310A) > 0 And _
  InStr("01,02,06,07", A0310B) > 0 And _
  InStr("2,3", A0310C) > 0 And _
  InStr("^", A0310D) > 0 And _
  InStr("99", A0310F) > 0 Then
  ISC = "NC"
'03/25/2010: Removed "99" from A0310B comparison below.
'05/25/2010: Removed "01" from A0310A comparison below.
ElseIf A0200 = "1" And _
  InStr("03,04,05", A0310A) > 0 And _
  InStr("01,02,03,04,05,06,07", A0310B) > 0 And _
  InStr("2,3", A0310C) > 0 And _
  InStr("^", A0310D) > 0 And _
  InStr("99", A0310F) > 0 Then
  ISC = "NC"
'03/25/2010: Removed "99" from A0310B comparison below.
ElseIf A0200 = "1" And _
  InStr("02,06", A0310A) > 0 And _
  InStr("01,02,03,04,05,06,07", A0310B) > 0 And _
  InStr("2,3", A0310C) > 0 And _
  InStr("^", A0310D) > 0 And _
  InStr("99", A0310F) > 0 Then
  ISC = "NQ"
ElseIf A0200 = "1" And _
  InStr("99", A0310A) > 0 And _
  InStr("01,02,03,04,05,06", A0310B) > 0 And _
  InStr("2,3", A0310C) > 0 And _
  InStr("^", A0310D) > 0 And _
  InStr("99", A0310F) > 0 Then
  ISC = "NP"
ElseIf A0200 = "1" And _
  InStr("99", A0310A) > 0 And _
  InStr("07", A0310B) > 0 And _
  InStr("2,3", A0310C) > 0 And _
  InStr("^", A0310D) > 0 And _
  InStr("99", A0310F) > 0 Then
  ISC = "NO"

' 04/29/2011: Added logic to handle A0310C=4
ElseIf A0200 = "1" And _
  InStr("01", A0310A) > 0 And _
  InStr("02,07", A0310B) > 0 And _
  InStr("4", A0310C) > 0 And _
  InStr("^", A0310D) > 0 And _
  InStr("99", A0310F) > 0 Then
  ISC = "NC"
ElseIf A0200 = "1" And _
  InStr("03,04,05", A0310A) > 0 And _
  InStr("02,03,04,05,07", A0310B) > 0 And _
  InStr("4", A0310C) > 0 And _
  InStr("^", A0310D) > 0 And _
  InStr("99", A0310F) > 0 Then
  ISC = "NC"
ElseIf A0200 = "1" And _
  InStr("02,06", A0310A) > 0 And _
  InStr("02,03,04,05,07", A0310B) > 0 And _
  InStr("4", A0310C) > 0 And _
  InStr("^", A0310D) > 0 And _
  InStr("99", A0310F) > 0 Then
  ISC = "NQ"
ElseIf A0200 = "1" And _
  InStr("99", A0310A) > 0 And _
  InStr("02,03,04,05", A0310B) > 0 And _
  InStr("4", A0310C) > 0 And _
  InStr("^", A0310D) > 0 And _
  InStr("99", A0310F) > 0 Then
  ISC = "NP"
ElseIf A0200 = "1" And _

```

```

InStr("99", A0310A) > 0 And _
InStr("07", A0310B) > 0 And _
InStr("4", A0310C) > 0 And _
InStr("^", A0310D) > 0 And _
InStr("99", A0310F) > 0 Then
ISC = "NO"

'SWING BED ISCs WHEN A0310D=0 (NOT CLINICAL CHANGE ASSESSMENT)
ElseIf A0200 = "2" And _
    InStr("99", A0310A) > 0 And _
    InStr("99", A0310B) > 0 And _
    InStr("0", A0310C) > 0 And _
    InStr("0", A0310D) > 0 And _
    InStr("01,12", A0310F) > 0 Then
    ISC = "ST"
ElseIf A0200 = "2" And _
    InStr("99", A0310A) > 0 And _
    InStr("01,02,03,04,05,06", A0310B) > 0 And _
    InStr("0", A0310C) > 0 And _
    InStr("0", A0310D) > 0 And _
    InStr("10,11", A0310F) > 0 Then
    ISC = "SP"
ElseIf A0200 = "2" And _
    InStr("99", A0310A) > 0 And _
    InStr("99", A0310B) > 0 And _
    InStr("0", A0310C) > 0 And _
    InStr("0", A0310D) > 0 And _
    InStr("10,11", A0310F) > 0 Then
    ISC = "SD"
ElseIf A0200 = "2" And _
    InStr("99", A0310A) > 0 And _
    InStr("01,02,03,04,05,06", A0310B) > 0 And _
    InStr("1", A0310C) > 0 And _
    InStr("0", A0310D) > 0 And _
    InStr("10,11", A0310F) > 0 Then
    ISC = "SP"
ElseIf A0200 = "2" And _
    InStr("99", A0310A) > 0 And _
    InStr("07", A0310B) > 0 And _
    InStr("1", A0310C) > 0 And _
    InStr("0", A0310D) > 0 And _
    InStr("10,11", A0310F) > 0 Then
    ISC = "SSD"
ElseIf A0200 = "2" And _
    InStr("99", A0310A) > 0 And _
    InStr("01,02,03,04,05,06", A0310B) > 0 And _
    InStr("2,3", A0310C) > 0 And _
    InStr("0", A0310D) > 0 And _
    InStr("10,11", A0310F) > 0 Then
    ISC = "SP"
' 04/29/2011: Added logic to handle A0310C=4
ElseIf A0200 = "2" And _
    InStr("99", A0310A) > 0 And _
    InStr("02,03,04,05", A0310B) > 0 And _
    InStr("4", A0310C) > 0 And _
    InStr("0", A0310D) > 0 And _
    InStr("10,11", A0310F) > 0 Then
    ISC = "SP"
ElseIf A0200 = "2" And _
    InStr("99", A0310A) > 0 And _
    InStr("07", A0310B) > 0 And _
    InStr("2,3", A0310C) > 0 And _
    InStr("0", A0310D) > 0 And _
    InStr("10,11", A0310F) > 0 Then
    ISC = "SOD"
' 04/29/2011: Added logic to handle A0310C=4
ElseIf A0200 = "2" And _
    InStr("99", A0310A) > 0 And _
    InStr("07", A0310B) > 0 And _
    InStr("4", A0310C) > 0 And _
    InStr("0", A0310D) > 0 And _

```

```

    InStr("10,11", A0310F) > 0 Then
    ISC = "SOD"
ElseIf A0200 = "2" And _
    InStr("99", A0310A) > 0 And _
    InStr("01,02,03,04,05,06", A0310B) > 0 And _
    InStr("0", A0310C) > 0 And _
    InStr("0", A0310D) > 0 And _
    InStr("99", A0310F) > 0 Then
    ISC = "SP"
ElseIf A0200 = "2" And _
    InStr("99", A0310A) > 0 And _
    InStr("01,02,03,04,05,06", A0310B) > 0 And _
    InStr("1", A0310C) > 0 And _
    InStr("0", A0310D) > 0 And _
    InStr("99", A0310F) > 0 Then
    ISC = "SP"
ElseIf A0200 = "2" And _
    InStr("99", A0310A) > 0 And _
    InStr("07", A0310B) > 0 And _
    InStr("1", A0310C) > 0 And _
    InStr("0", A0310D) > 0 And _
    InStr("99", A0310F) > 0 Then
    ISC = "SS"
ElseIf A0200 = "2" And _
    InStr("99", A0310A) > 0 And _
    InStr("01,02,03,04,05,06", A0310B) > 0 And _
    InStr("2,3", A0310C) > 0 And _
    InStr("0", A0310D) > 0 And _
    InStr("99", A0310F) > 0 Then
    ISC = "SP"
' 04/29/2011: Added logic to handle A0310C=4
ElseIf A0200 = "2" And _
    InStr("99", A0310A) > 0 And _
    InStr("02,03,04,05", A0310B) > 0 And _
    InStr("4", A0310C) > 0 And _
    InStr("0", A0310D) > 0 And _
    InStr("99", A0310F) > 0 Then
    ISC = "SP"
ElseIf A0200 = "2" And _
    InStr("99", A0310A) > 0 And _
    InStr("07", A0310B) > 0 And _
    InStr("2,3", A0310C) > 0 And _
    InStr("0", A0310D) > 0 And _
    InStr("99", A0310F) > 0 Then
    ISC = "SO"
' 04/29/2011: Added logic to handle A0310C=4
ElseIf A0200 = "2" And _
    InStr("99", A0310A) > 0 And _
    InStr("07", A0310B) > 0 And _
    InStr("4", A0310C) > 0 And _
    InStr("0", A0310D) > 0 And _
    InStr("99", A0310F) > 0 Then
    ISC = "SO"

'SWING BED ISCs WHEN A0310D=1 (CLINICAL CHANGE ASSESSMENT)
ElseIf A0200 = "2" And _
    InStr("99", A0310A) > 0 And _
    InStr("01,02,03,04,05,06,07", A0310B) > 0 And _
    InStr("0,1,2,3", A0310C) > 0 And _
    InStr("1", A0310D) > 0 And _
    InStr("10,11,99", A0310F) > 0 Then
    ISC = "SP"
' 04/29/2011: Added logic to handle A0310C=4
ElseIf A0200 = "2" And _
    InStr("99", A0310A) > 0 And _
    InStr("02,03,04,05,07", A0310B) > 0 And _
    InStr("4", A0310C) > 0 And _
    InStr("1", A0310D) > 0 And _
    InStr("10,11,99", A0310F) > 0 Then
    ISC = "SP"

```

```
'ALL COMBINATIONS NOT CONSIDERED ABOVE ARE INVALID
Else
  ISC = "--"
End If

'Set return value equal to the value of ISC
GetIsc = ISC

End Function
```