Chapter 7: Edit Errors

Computerized edits are automated processes that test the validity of data fields. The following types of edits are currently integrated into SEER*DMS:

- The **SEER Edits** cover fields submitted to SEER. SEER*DMS uses the SEER Edits Engine, a Java software module that is also used by the SEER*Edits and SEER*Abs software.
- The **SEER Extended Edits** were developed by the NCI to extend edits beyond those fields sent to the SEER Program. The extended edits are available to all SEER*DMS registries and validate fields that are not required to be transmitted to SEER.
- **NPCR, NCDB, and NCFD Edit Sets** – Edits for the National Program of Cancer Registries (NPCR), National Cancer Database (NCDB), and NAACCR Call for Data (NCFD) are maintained and distributed in the NAACCR GenEDITS software. Many, but not all, of the NPCR, NCDB, and NCFD edits that are available in the GenEDITS software are also available in SEER*DMS. The SEER*DMS development team wrote a compiler that translates the GenEDITS metafile into the Groovy scripting language. The Groovy versions of the edits are deployed with SEER*DMS. An automatic translation is not possible for all edits; therefore, some NPCR, NCDB, and NCFD edits may be excluded. Updated versions of the edits will be deployed periodically.
- **Registry edits** are defined and maintained by registry staff.
- **System edits** are edits to enforce database constraints.

SEER, SEER Extended, the translated NCDB, NPCR, and NCFD edits, and all system edits are maintained and deployed by the SEER*DMS development team. Registry staff are responsible for defining and maintaining registry-specific edits that are not system edits. The SEER*DMS Edits Manager provides an interface for adding, modifying, and deleting registry edits.

A patient set that is in the workflow will not exit the workflow until all edits are cleared. If an edit failure is caused by a change made via mass change or ad hoc editing, a Resolve Patient Set Errors task will be created. However, there may be patient sets with edit errors that are not in the workflow. This would happen if a new or modified edit were executed across the full database using the Patient Set Edits system task. A system report to monitor edit failures is auto-generated when the edits system task is executed. This report (RPT-064A) can be used at any time to track and manage edit failures.

Edits are executed each time a user opens, validates, or saves a record or patient set in SEER*DMS. Edits are also executed whenever an automated process updates a patient set (this includes updates via Mass Change imports). The *Patient Set Edits System Task* can be used to execute the edits on all patient sets or a cohort of patient sets. The primary purpose of this task is to apply new or modified edits to all patient sets in the database.

In this chapter, you'll learn about

- Understanding Edits in SEER*DMS
- Viewing Edit Documentation
- Using the Edits Help Page
- Defining and Maintaining Registry Edits
- System Task to Execute Edits in Patient Sets
- System Reports Related to Edits
- Color Codes Used in SEER*DMS Editors

Understanding Edits in SEER*DMS

The components of SEER*DMS edits are described below.

Edit ID

A unique identifier assigned to each edit. You must specify an ID for a new registry edit (1-50 alphanumeric characters may be used).

Guidelines for specifying an ID for a registry edit:

- Use your registry's two-character abbreviation as a prefix for registry-defined edits (AK, CN, CT, DT, HI, IA, LA, NM, SE, UT, GA, NJ).
- If you are creating an edit designed to replace or supplement a SEER Edit, the ID should consist of your registry's prefix followed by the SEER Edit ID (e.g., DT_IF29, IA_IF02).

Active Flag

An edit can be turned on or off by toggling its Active flag. Inactive edits will not fail for any data. All failures related to an edit are removed from the database when an edit is deactivated. The database is updated when the edit is saved. All other changes to an edit (including activating an edit) are not applied to the database until the Patient Set Edits system task is executed.

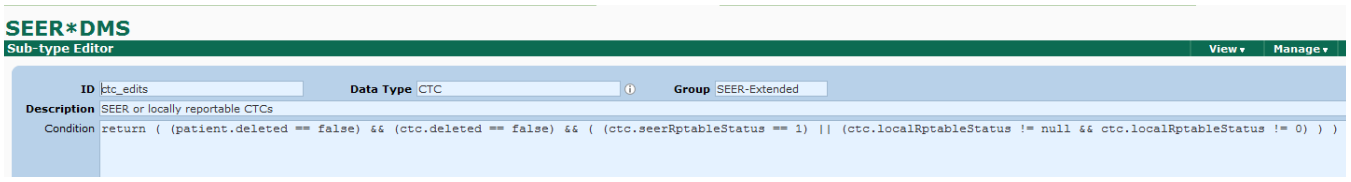
Group

In SEER*DMS, edits are organized into groups:

1. SEER Edits validate fields submitted to SEER.
2. SEER-Extended Edits are defined and maintained by SEER for the SEER*DMS registries. These validate fields supported in SEER*DMS, but not required to be transmitted to SEER.
3. Registry Edits are defined and maintained by registry staff. These would include edits to validate registry-specific fields, edits to implement validation not covered by SEER edits, and registry-specific versions of SEER edits (the registry may decide to vary the logic of a SEER edit by creating a registry-defined edit and disabling the SEER edit).
4. Database Constraints are edits that enforce the database integrity of fields common to all SEER*DMS registries
5. Database Constraints Registry edits enforce database integrity of registry-specific fields. The SEER*DMS development team creates these edits, if necessary, when registry-specific fields are added to the database.
6. NPCR Edits – National Program of Cancer Registries edits that could be translated from the GenEDITS source code into the Groovy scripting language required by SEER*DMS. An edit will be available if the source code could be successfully translated and the edit is compatible with SEER*DMS data structures.
7. NCDB Edits – National Cancer Database edits translated from the GenEDITS source code.
8. NCFD Edits - NAACCR Call for Data edits translated from the GenEDITS source code.

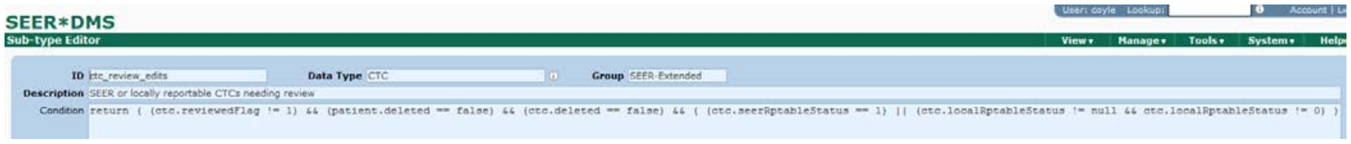
Data Type and Sub-type

Data Type and Sub-type define the data to which the edit is applied. Data types are the database entities. Sub-types are subsets of the database structures defined as a Boolean statement and implemented in Groovy code. For example, CTC is the Data Type for the ADDR_DX_STATE edit. The sub-type is SEER or locally reportable CTCs. The definition for this sub-type is shown below.



ID ctc_edits
Data Type CTC
Group SEER-Extended
Description SEER or locally reportable CTCs
Condition
```
return ( (patient.deleted == false) && (ctc.deleted == false) && ( (ctc.seerRptableStatus == 1) || (ctc.localRptableStatus != null && ctc.localRptableStatus != 0) ) )
```

Sub-types may be used to limit the execution of edits based on criteria (e.g. reportable status) or to attach an edit to a review flag (e.g., edits may be run against “CTCs needing review”).



ID ctc_review_edits
Data Type CTC
Group SEER-Extended
Description SEER or locally reportable CTCs needing review
Condition
```
return ( (ctc.reviewedFlag != 1) && (patient.deleted == false) && (ctc.deleted == false) && ( (ctc.seerRptableStatus == 1) || (ctc.localRptableStatus != null && ctc.localRptableStatus != 0) )
```

NAACCR Lines are special data types defined for the SEER Edits Engine. The SEER Edits Engine can be used to validate data stored in any data format (relational database, text files, data entry forms, etc). In SEER*DMS, the data are stored in a relational database and converted to a NAACCR Line for validation by edits maintained in the SEER Edits Engine.

Guidelines for defining Sub-types for registry edits:

- Record Edits: Most record edits are applied to all records. You may need to create a sub-type if you are defining an edit that will only be applied to certain record types.
- Patient Set Edits: Based on the fields used in the edit’s logic, define or use a sub-type related to the database table at the lowest level of the patient set hierarchy.
    - Identify the database table for each field used in the logic of the edit.
    - Refer to the Patient Set E-R Diagram provided in the Database Documentation section of the SEER*DMS Web Portal (see Database Diagrams) .
    - Determine the lowest level table used in the edit’s logic.
    - Select a sub-type related to that table.
    - For example, an edit comparing age at diagnosis and age at birth uses the CTC and Patient tables. The CTC table is at a lower level in the patient set hierarchy than the patient table. Therefore, you should select an entity related to the CTC table.
- If your edit requires an entity that is not listed, you may create one or submit a request to the SEER*DMS Technical Support team.

Message

The edit message is displayed in the patient set or record editor when the edit fails. The Edit ID and Message are shown when you hover over a field involved in the failed edit. The edit messages are also displayed in all edit reports. The maximum length for an edit message is 500 characters.

Guidelines for defining Messages for registry edits:

- The edit message should reflect the logic of the edit and should be written in language that is easily understood by data editors.
- It is recommended that you include “review required” in the message text if the edit is associated with a review flag.

Severity

Each edit is assigned a severity level of low, moderate, high, or critical. The severity level is used to determine whether the data can be saved, whether a manual task is required, and to prioritize the edits for the user performing editing tasks.

Record edits are assigned a high severity level if the data item is required in order to screen the record for reportability or special studies. Therefore, a record edit with a high severity level will trigger a Resolve Record Errors task. If a record only has fields with low or moderate edit errors, the fields are flagged but the record moves forward in the workflow and is screened for reportability. If a record has one or more fields with high or critical edit errors, the record is sent to a manual Resolve Record Errors task for review prior to reportability screening.

Edits with a critical error level must be resolved in order to save the record or patient set. The critical error level is typically reserved for edits that check values to determine if they violate database constraints. For example, a critical edit error would occur if an "A" were entered in a field that requires a numeric value. *Note: It is technically possible to configure the edits to use the critical error level for other purposes. However, it is recommended that you only assign the critical error level to edits that check for database constraints.*

Guidelines for setting the severity level of a registry edit:

- You should rarely define an edit as critical. Typically, the SEER*DMS development team will define a critical edit, if necessary, when a registry-specific data field is added to the database. Remember that your editing staff will not be able to save their changes if the changes generate a critical error.
- Record Edits:
  - Set the severity level to High if you want SEER*DMS to create a Resolve Record Errors task when a record fails this edit.
  - Set the severity level to High if the edit is tied to the record reviewed flag for the purpose of initiating follow-back. A manual Resolve Record Errors task will provide a user with the opportunity to create the follow-back, set the reviewed flag, and save the record. The record would then move forward in the workflow.
  - If you want the edit to flag fields as having a problem but do not want a Resolve Record Errors task created, set the severity level to either low or moderate. Registry policy dictates when to use low versus moderate.
- Patient Set Edits:
  - Registry policy dictates whether to assign a severity of low, moderate, or high to a patient set edit.
  - For SEER and SEER*DMS edits, the low severity level typically indicates that a review flag will over-ride the edit.
  - The severity levels of edits do not affect the patient set workflow. If a patient set is saved with edit failures in a Visual Edit Patient Set or Consolidate task it will move to a Resolve Patient Set Errors task, regardless of the severity level of the failing edits.

Dependencies

A dependency prevents an edit from executing unless the data passes another edit. Typically, a dependency is added to an edit to ensure that a valid value is provided for a required field. For example, several SEER edits require age at diagnosis. These edits are not executed if the "*age_at_diagnosis*" edit fails.

Two dependency lists are shown for each edit. An edit is not executed if any of the edits listed in its "Depends on" section fail. Similarly, if the edit fails then none of the edits in the "Depended on by" section will be applied.

To view examples, review the dependencies for the following SEER edits: Age_at_diagnosis (Age at diagnosis is not valid), IF118 (Age, Primary Site, Histology conflict), IF13 (Age, Birth Date and Date of Diagnosis conflict).

Dependencies can only be defined between edits within the same group. For example, a registry edit can only depend on registry edits and cannot depend on SEER edits.

Groovy Code

SEER*DMS edits are implemented in Groovy, a scripting language for the Java platform. The Internet has several Groovy references including the Groovy home page at *groovy.codehaus.org*. You do not need to be fluent in Groovy to add or modify edits. SEER*DMS edits use a small subset of the Groovy syntax. A working knowledge of regular expressions and Groovy logic statements are all that is needed to add and modify edits in SEER*DMS. To define a new edit, it is recommended that you copy an existing edit and use it as a template.

Guidelines for writing the Groovy code for a registry edit:

- An edit error is triggered if the code returns FALSE for the record or patient set. The edit passes if the code returns TRUE.
- Reference a field by the alias for the entity and the property name. This is listed as Syntax for Edit Source Code in the Properties tab.
- Use the Groovy code of a similar edit as a template.
- If you copy the Groovy code of a SEER, SEER*DMS, or SEER Extended edits, verify that an appropriate entity is available. If your edit requires an entity that is not listed, create a sub-type or request support via the SEER*DMS Technical Support Squish project.
- Create unit tests to verify the logic of the edit.

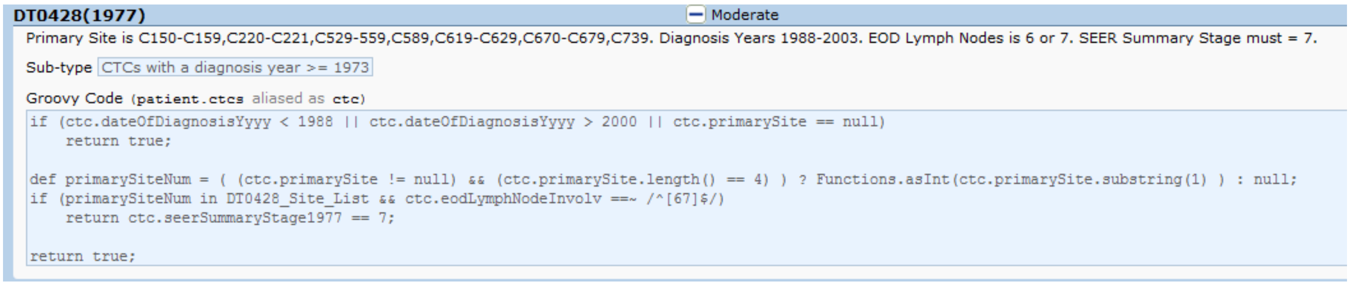
Contexts

A context is a Java naming system. In SEER*DMS, contexts are used to define arrays, hash tables, and functions used by the edits. For example, there are a large number of contexts defined for the SEER*Edits. Primarily, these represent data tables required by the SEER Edits logic. Registry staff may add or modify contexts by selecting **System > Edits > Context**. A working knowledge of Groovy is required to write Contexts.

A Context is directly available to edits within the Context's group. However, edits in one group may reference Contexts in other groups. You must specify the group in a call to the getContext method. In this example, a context (*repYearByRegistryId*) from another group (SEER) is used:

def yearByRegistry =
Functions.asInt(Functions.getContext("seer", "repYearByRegistryId")[registryId]);

To use a context defined for registry edits, you simply need to enter the context name. In this example, DT0428_Site_List is a registry-defined context that defines a list of sites.



DT0428(1977) Moderate

Primary Site is C150-C159,C220-C221,C529-559,C589,C619-C629,C670-C679,C739. Diagnosis Years 1988-2003. EOD Lymph Nodes is 6 or 7. SEER Summary Stage must = 7.

Sub-type CTCs with a diagnosis year >= 1973

Groovy Code (patient.ctcs aliased as ctc)

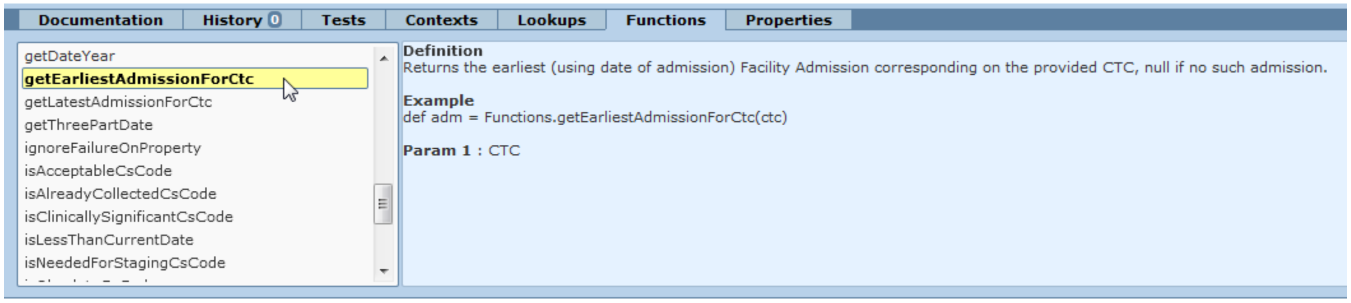
```
if (ctc.dateOfDiagnosisYyyy < 1988 || ctc.dateOfDiagnosisYyyy > 2000 || ctc.primarySite == null)
    return true;

def primarySiteNum = ( (ctc.primarySite != null) && (ctc.primarySite.length() == 4) ) ? Functions.asInt(ctc.primarySite.substring(1) ) : null;
if (primarySiteNum in DT0428_Site_List && ctc.eodLymphNodeInvolv ==~ /^[67]$/)
    return ctc.seerSummaryStage1977 == 7;

return true;
```

Lookups

The lookups used by edits are queries that verify values against tables in the database. These are primarily used to query lookup tables in the database (tables with the lkup prefix). Typically, the edit lookups simply check to see if a code exists in the database table, but other logic may be incorporated. The full query for each lookup can be seen in the list of Lookups shown on the Definition tab of the Edit Editor. Registry staff may not add or modify lookups in the current version of SEER*DMS.

Functions

Functions written and maintained by the SEER*DMS development team may be used in registry edits. The function declaration and examples are provided for each function as shown below.



getDateYear
getEarliestAdmissionForCtc
getLatestAdmissionForCtc
getThreePartDate
ignoreFailureOnProperty
isAcceptableCsCode
isAlreadyCollectedCsCode
isClinicallySignificantCsCode
isLessThanCurrentDate
isNeededForStagingCsCode

Definition
Returns the earliest (using date of admission) Facility Admission corresponding on the provided CTC, null if no such admission.

Example
def adm = Functions.getEarliestAdmissionForCtc(ctc)

Param 1 : CTC

To find an example of an edit using this function, search for “getEarliestAdmissionForCtc” on the Edits help page. To call this function, you would use this statement:

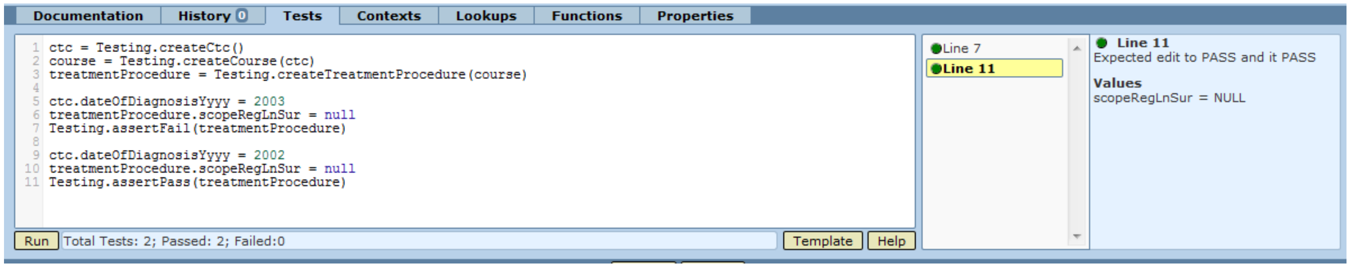
def earliest = Functions.getEarliestAdmissionForCtc(ctc)

Documentation

The documentation section contains descriptive text that is more comprehensive than the edit’s message. If available, this documentation is displayed in the edit popup shown in the patient set and record editor. Documentation provided by the SEER program is included for the SEER Edits. Registry staff are responsible for writing and maintaining documentation for registry edits.

Unit Tests

A set of unit tests may be defined for each edit. The use of unit tests is strongly recommended for inter-field edits maintained by the registry. On the Tests tab, you may click Template to create source code for a unit test. Enter a value for the properties used by the edit and uncomment those lines. For some tests, you may not need to specify values for all fields. Uncomment the *assertFail* line if the edit should fail for the defined values. Uncomment the *assertPass* line if you specified values that should pass the edit.



```
 1 ctc = Testing.createCtc()
 2 course = Testing.createCourse(ctc)
 3 treatmentProcedure = Testing.createTreatmentProcedure(course)
 4
 5 ctc.dateOfDiagnosisYyyy = 2003
 6 treatmentProcedure.scopeRegLnSur = null
 7 Testing.assertFail(treatmentProcedure)
 8
 9 ctc.dateOfDiagnosisYyyy = 2002
10 treatmentProcedure.scopeRegLnSur = null
11 Testing.assertPass(treatmentProcedure)
```

Total Tests: 2; Passed: 2; Failed:0

● Line 7
● Line 11

● **Line 11**
Expected edit to PASS and it PASS

Values
scopeRegLnSur = NULL

Multiple tests may be defined. To create a second unit test: you may create and modify a new template; or you may copy-and-paste the lines of code that set values and copy the appropriate assertion. Modify the lines in the new text so that the appropriate values are set for each field.

Viewing Edit Documentation

SEER*DMS provides four ways to view documentation related to edits:

- The SEER*DMS editor allows you to view documentation for edits that are failing in the record or patient set. The Edits are displayed on a tab within the right panel. The full documentation and Groovy code can be viewed by clicking the information icon (you may toggle the view between definition and source code).
- All system users have access to the Help menu which includes an Edits section. You may use the Edits Help page to search for an edit by edit ID, group, severity level, and/or search text (see the *Using the Edits Help Page* section of this chapter for instructions).
- If you have the *edits_manager* permission you may use the Edits Manager to find, review, and modify edits. The documentation, code, and unit tests can be viewed for each edit.
- If you have the *system_administration* permission, you can search the XML configuration files in which the SEER, SEER Extended, and system edits are defined. Refer to the *System Administration Page* section of Chapter 27 for instructions to access the system files. The edit system files are named according to the edit group. Once you select the XML file, use the Firefox search tools to find a particular edit.

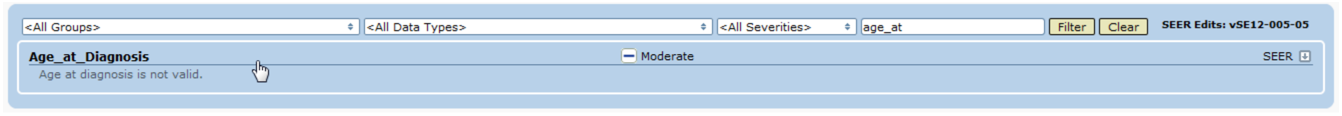
Using the Edits Help Page

Requires system permission: *none* (all users have access to the help pages)

To search the Edits Help page for documentation related to an edit:

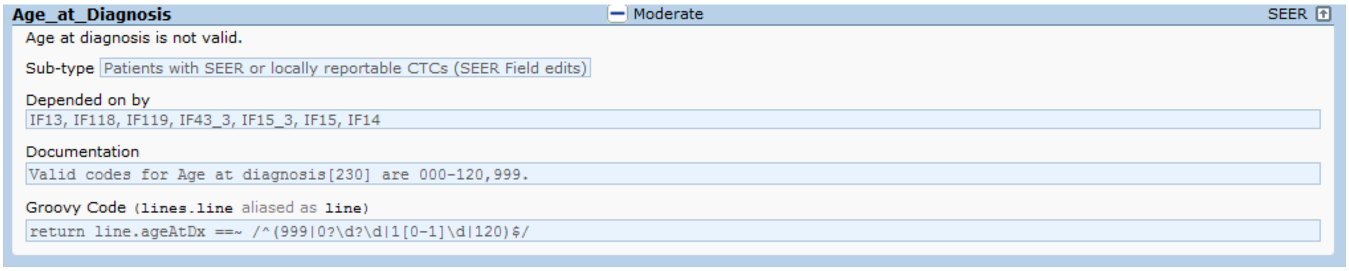
1. Select **Help** > **Edits**. By default, all edits will be listed.
2. Use the drop-down filters to search by edit group, data type (entities), and severity level.
3. Enter text in the *keywords* box to search for text in the edit message, ID, or Groovy source code.
4. Click **Filter** to apply the search criteria; or **Clear** to reset the filter to the default values.

The following example shows the result of using the filter to find Edit Group = SEER, Data Type = All, Severity Level = All, and search text = *age_at*.



Age_at_Diagnosis — Moderate — SEER
Age at diagnosis is not valid.

Click the edit's ID to expand its help section, as shown below. The edit's Message is listed first, followed by the Sub-type, Dependencies, Documentation, Groovy Code, and History (if available). Each component is described in the *Understanding Edits in SEER*DMS* section of this chapter.



Age_at_Diagnosis — Moderate — SEER
Age at diagnosis is not valid.

Sub-type: Patients with SEER or locally reportable CTCs (SEER Field edits)

Depended on by: IF13, IF118, IF119, IF43_3, IF15_3, IF15, IF14

Documentation: Valid codes for Age at diagnosis[230] are 000-120,999.

Groovy Code (lines.line aliased as line)

```
return line.ageAtDx ==~ /^(999|0?\d?\d|1[0-1]\d|120)$/
```

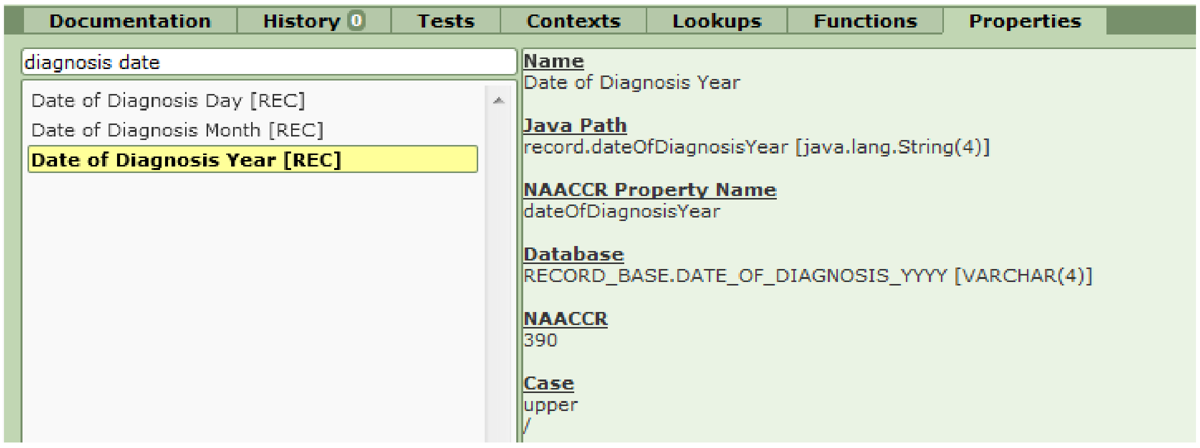
Defining and Maintaining Registry Edits

Requires system permission: *edits_manager*

The instructions below provide an overview of the steps required to add, delete, or modify an edit. Refer to the *Understanding Edits in SEER*DMS* section of this chapter for specific guidelines when selecting values for each component of the edit.

To create or modify a registry edit:

1. Select **System > Edits > Rules**.
2. If you are creating a new edit, select **Actions > Add Edit**. If you are modifying an existing edit, use the filters to find the edit and click the edit's **ID**.
3. Set the edit components; guidelines are provided in Understanding Edits in SEER*DMS.
   a. Enter or modify the **Edit ID**.
   b. Select one item listed in the **Data Sub-type** drop-down list.
   c. Select the **Severity** level that you wish to assign to this edit.
   d. Enter or modify the **Message** text.
   e. An edit is not executed if any edit listed in the **Depends on** section fails. To add or remove an edit in the Depends On list:
      i. Click to expand the **Depends On** drop-down list.
      ii. Use the filter to search for the edit. Dependencies can only be defined between edits that use compatible Data Types, therefore, no edits will be available in the drop-down until you specify the Data Sub-type.
      iii. Check or uncheck the box next to the **Edit ID**.
4. In the **Definition** box, enter or modify the **Groovy** source code that defines the logic of the edit. Use the tabs at the bottom of the page to enter Documentation, view the History of changes, define unit Tests; and to view the available Contexts, Lookups, Functions, and Properties.
   a. On the **Documentation** tab, enter detailed documentation related to the new edit or modify the documentation to reflect the changes that you made to the logic.
   b. Please refer to the Understanding Edits in SEER*DMS section of this chapter for instructions on using **Contexts**, **Lookups**, and **Functions** in registry edits.
   c. Use the **Properties** tab to search for a field's documentation. You must use the Java Path to reference a field in the source code of any edit.


Documentation
History 0
Tests
Contexts
Lookups
Functions
Properties
diagnosis date
Date of Diagnosis Day [REC]
Date of Diagnosis Month [REC]
Date of Diagnosis Year [REC]

Name
Date of Diagnosis Year

Java Path
record.dateOfDiagnosisYear [java.lang.String(4)]

NAACCR Property Name
dateOfDiagnosisYear

Database
RECORD_BASE.DATE_OF_DIAGNOSIS_YYYY [VARCHAR(4)]

NAACCR
390

Case
upper

   d. Create unit **Tests** to verify the logic of the edit. Click the Help button on the Tests tab for instructions.

e. The **History** tab will show the date of each change and the comment entered by the programmer who made the change.

5. Click **Save** to exit and save your changes.

To delete a registry edit:

1. Select **System > Edits > Rules**.
2. Use the filters to find the edit as described in the *Using the Edit Manager* section of this chapter. To search for registry-defined edits, set the Group filter to your registry's name. (SEER, SEER Extended, SEER*DMS, and SEER*DMS Registry edits cannot be deleted.)
3. Click the edit's **ID**.
4. Review the list of edits that depend on this edit (expand the Depended on By list to view edit messages). Determine whether this edit must be retained in order for those edits to function properly; or if the listed edits require modifications due to this edit being deleted.
5. If you determined that the edit can be deleted, click the **Delete** button.

System Task to Execute Edits in Patient Sets

All edits are executed each time a patient set is opened, validated, or saved in the SEER*DMS editor. Edits are also executed whenever an automated process updates a patient set (this includes updates via Mass Change imports). Use the Patient Set Edits system task to re-execute the edits on patient sets in the database. You may run the edits on all patient sets or on a cohort defined by year of diagnosis. Use this task to ensure that new or modified edits are evaluated.

A polisher is a system utility that derives, calculates, or assigns data field values. For example, polishers are used to derive collaborative stage variables; assign census tract based on address; and calculate the age at diagnosis based on date of birth and date of diagnosis. When a patient set is opened, saved, or validated, a polisher will be executed if the value of a related data item changes. Occasionally, you may need to execute a polisher on a large number of patient sets. The Patient Set Edits task enables you to run selected polishers as well as the edits.

Polishers are assigned to "classes". Classes are simply categories used in SEER*DMS to define when the polishers are executed during various system processes. A polisher may belong to multiple classes. The Polishers help page lists the polishers and the classes to which they belong.

The Patient Set Edits task has an option to execute the Edit Polishers. Edit polishers are polishers in the "Pre-Edits" classes (the polishers in these classes are listed on the screen when you open the Patient Set Edits system task). These are polishers that need to be run with the Patient Set Edits task during the initial migration of data to SEER*DMS. The edit polishers rarely need to be run as a group after deployment.

A polisher may need to be executed with the Patient Set Edits task when a new version of SEER*DMS is released that includes a change to that polisher. The Patient Set Edits task allows you to select one to three "extra" polishers. All of the standard polishers (polishers in the "Validation" class) can be executed with the Patient Set Edits task.

To re-execute the edits for some or all patient sets in the database:

1. Click **System > Tasks**.
2. Click the **Patient Set Edits** link.
3. To limit the edits to data by year of diagnosis, enter a **Start Year** and **End Year**. Patient sets with a diagnosis date during this time period will be considered.
4. To include data with unknown year of diagnosis, set **Include Unknown Year** to *Yes*.

5. If you wish to execute all polishers in the Pre-Edit and Post-Edit classes, set **Run Edit Polishers** to *Yes*. It is recommended that this option be set to *No* unless there is a specific need related to the transitioning of data into SEER*DMS.
6. You may execute one to three individual polishers with the Patient Set Edits using the **Extra Polisher** drop-down menus. These are polishers in the pat-validation class (as listed on the Polishers help page).
7. You may enter text related to this task in the **Comment** field. The comment for the last execution of the task is stored in the database (utility_history table).
8. Click **Start**.

The edits will be re-evaluated for each patient set in the cohort. In order to avoid creating an inordinate number of worklist tasks, a Resolve Patient Set Errors task will *not* be created for each patient set with an edit error. If the logic of a new or modified edit is implemented incorrectly, it could erroneously create an edit error for a large number of patient sets. Therefore, you must use reports rather than tasks to identify the patient sets with errors and to evaluate the error levels in the patient set data. Two system reports, RPT-064A and RPT-064B, are available for identifying the edit errors that were triggered and the patient sets that are involved:

System Reports Related to Edits

Report ID	Title	Description
RPT-064A	Frequency of Edit Errors in Patient Set Data	Use this report to evaluate the error levels in the Patient Set data
RPT-064B	Patient Sets with Edit Errors	A listing of Patient Sets with a failure related to a particular edit or a sample for all failing edits.
RPT-064C	SEER*DMS Edits	ID, Message, Severity, and Group of all edits available in SEER*DMS.
RPT-064D	Detailed Listing of SEER*DMS Edits	Lists all components of edits, including the Groovy code, entities, history, and documentation.
RPT-064E	Number of Edit Failures per Patient Set	Lists the Patient Set ID and the number of failed edits for each Patient Set with edit errors.

Color Codes Used in SEER*DMS Editors

The SEER*DMS record and patient set editors display data fields in different colors to highlight fields that contain errors, and fields that have been modified but not validated or saved. Although you can readily see the fields that require attention on data pages, it is recommended that you review the list of errors and related data fields on the Edit Errors page prior to making changes to a record or patient set. Refer to the chapter related to your specific task for further instructions.

If two or more edits are related to a field, the edit with the highest severity level will determine the color.

1981	**White** indicates that the field does not have an error and has not been changed since the last save.
2010	**White framed with a color** indicates that your cursor is at that field. The color used to frame the box varies by color scheme (an orange frame is used in the green color scheme shown here).
XXXX	**Dark red with white text** indicates that the field triggered a **Critical Error**. The value can not be stored in the system due to database constraints. Critical Errors will only be seen in patient sets, since database constraints are not applicable to record data.
0000	**Medium red with black text** indicates that the field triggered an error with a **High** severity level. In patient set data, the value may not be valid for this field or an inter-field edit may have detected a conflict with other fields. In record data fields, this indicates that an error was detected that must be resolved prior to screening. The record will not move past the Resolve Record Errors task until all errors with a severity level of high are resolved.
9999	**Light red** indicates that the field has an error with a **Low** or **Moderate** severity level. The value may not be valid for this field or an inter-field edit may have detected a conflict with other fields. Low and moderate errors should not be resolved when editing records, these editing tasks should be performed when consolidating the patient data.
1980	**Orange** indicates that a field with an error was modified, but the change has not been validated against the SEER and local edits. Fields are validated when the record is saved or the Validate button is clicked. Once validated, this field will turn to yellow if the error was corrected or to a shade of red if the field still contains an error.
1990	**Yellow** indicates that the field that did not have an error and was changed. The patient set or record has not been saved since the change was made. Once validated, this field will turn to a shade of red if there is an error.
1980	**Blue with a black border** indicates that the field was changed by a Polisher.
0000	**Blue with no border** indicates that the field is read-only and was changed by a Polisher. *Note: Some fields are modified when another field is changed.*
DOE, JOHN	Read-only fields that have not been changed are shown in a lighter shade (the color is determined by the color scheme that you are using).