# 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. The SEER*DMS development team wrote a compiler that translates GenEDITS metafiles into the Groovy scripting language. The Groovy versions of the edits are deployed within SEER*DMS.
- **Registry edits** are written to validate registry-specific fields or enforce registry-specific rules for standard fields.
- **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. The SEER*DMS Edits Manager provides an interface for adding, modifying, and deleting registry edits. Registry staff are responsible for defining and maintaining registry-specific edits that are not system edits. The SEER*DMS technical support team will provide assistance.

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. The Edits Dashboard and a system report (RPT-064A) can be used to monitor edits 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.

Edits are stored in the database in the validator_rule table. There are several tables in the database related to edits. If you are interested in the database structure, review the sample SQL in the SQL Data Search. There are several samples related to edits.

In this chapter, you'll learn about

- Methods of Viewing Edit Documentation
- Viewing Documentation with the Edits Manager
- Defining and Maintaining Registry Edits
- Understanding Edits in SEER*DMS
- Edits Manager Tabs
- Edit Sets
- Contexts

- System Task to Execute Edits in Patient Sets
- System Reports Related to Edits
- Color Codes Used in SEER*DMS Editors

## Methods of Viewing Edit Documentation

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

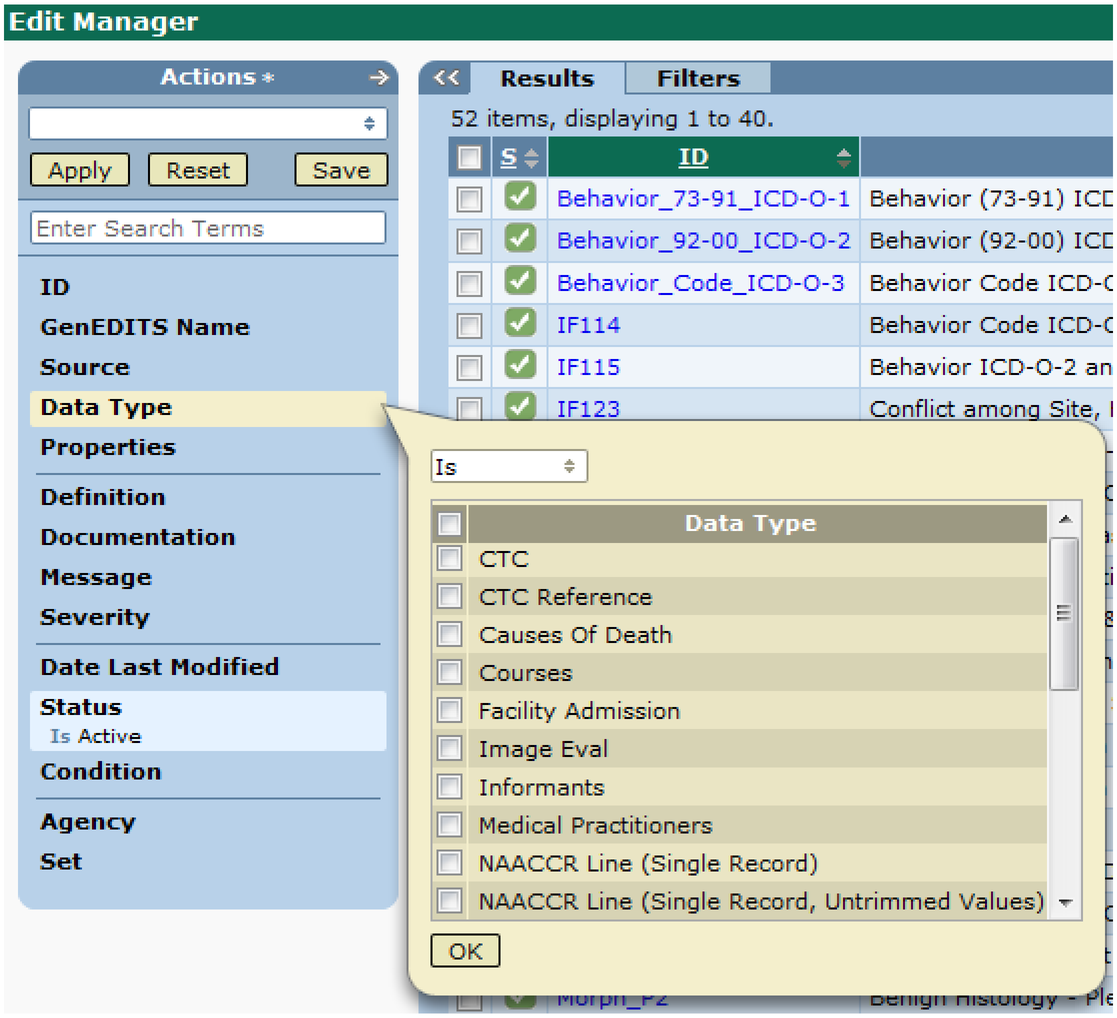
- The SEER*DMS editor allows you to view documentation for edits that are failing in the record or patient set. The Edits are displayed on a tab within the right panel. The full documentation and Groovy code can be viewed by clicking the information icon (you may toggle the view between definition and 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.

## Viewing Documentation with the Edits Manager

Requires system permission: *none* (all users have access to the Edits Manager)

*To search the Edits Manager:*

1. Select **System > Edits Manager > Edits** or **Help** > **Edits**. By default, all edits will be listed.
1. Use the filters to narrow down your search.
2. Enter text in the Search Terms box to search for text in the ID.
3. Click **Filter** to apply the search criteria; or **Clear** to reset the filter to the default values.
4. Click on the Edit ID to view the Definition and Documentation related to the Edit.



## 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. See the Understanding Edits in SEER*DMS section of this chapter for more information on each component.

1. Select **System > Edits Manager > Edits** and click the arrow on the Actions menu. 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**.
2. Set the **ID**, **Data Type**, **Severity** and **Message**. **GenEDITS** and **Agency** are not necessary.
3. Set the **Depends on** and **Depended on by** drop downs.
4. In the **Definition** box, enter or modify the **Groovy** source code that defines the logic of the edit.
5. Use the tabs at the bottom of the page to enter Documentation, view the History of changes, define unit Tests. You can also view the available Contexts, Lookups, Functions, Conditions, and Properties.
6. Click **Save** to exit and save your changes.

*To delete a registry edit:*

1. Select **System > Edits Manager > Edits**.
2. Click the edit's **ID**.
3. 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.
4. If you determined that the edit can be deleted, click the **Delete** button.

*To Inactivate an Edit*

An edit can be set to inactive by accessing the edit through the Edits Manager (**System > Edits Manager > Edits**) and setting the condition to **Inactive**. This means the edit will no longer be executed, and it will be cleared from records and patient sets for which it had already been triggered. You cannot set System Edits (Database Constraints) to Inactive.

You can also toggle between Active/Inactive by selecting the Edit in the Edits Manager and by clicking **Modify** in the Actions Menu.

## Understanding Edits in SEER*DMS

The components of the Edit Editor are described below.

### Edit ID

The edit ID is a unique identifier assigned to each edit. You must specify an ID for a new registry edit (1-50 alphanumeric characters may be used; underscore, hyphen, and parentheses are also allowed).

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)

### Data Type

Data Types are database entities that define the data to which the edit is applied. You must select the Data Type before setting Conditions. NAACCR Lines are special data types defined for the SEER*Utils Edits Engine. The SEER*Utils Edits Engine can be used to validate data stored in any data format (relational database, text files, data entry forms, etc). This engine is used by the SEER*Abs, SEER*Edits, and SEER*DMS software. SEER*Abs converts data from data entry forms into a NAACCR line for validation. SEER*Edits loads data from text files to create NAACCR Lines. In SEER*DMS, the data are stored in a relational database but converted to a NAACCR Line for validation by edits maintained in the SEER*Utils Edits Engine.

## 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.

Except for critical edits, 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.

## Agency

Agency is a field in the GenEDITS metafile. It is retained in SEER*DMS to be consistent with GenEDITS. The agency is null for edits created in SEER*DMS, therefore, it is null for all registry edits. The agency is set to IMS for edits written specifically for SEER*DMS and maintained by IMS.

These are edits that have a source of SEER Extended, Database Constraints, or Database Constraints Registry.

### Status

Status is calculated based on the edit's conditions, sets, and dependencies.

- Conditions - Status is inactive if the rule is associated with one of the two system conditions that prevent a rule from executing. These conditions are read only and SEER*DMS specifically checks for them by ID. The "Inactive (Patient Set Edits)" deactivates edits for any patient set field. The "Inactive (Record Edits)" condition deactivates record edits. When showing the status for an edit, SEER*DMS does not evaluate all of the edit's conditions. An edit could have a condition that could never be true, but SEER*DMS could show it as active.
- Sets - If the rule is not a member of at least one active set then status will be inactive.
- Dependencies - Status will be inactive if the edit depends on an edit that is inactive.

### Sets

A set is a group of edits. An edit will not be executed unless it is a member of at least one active edit set. An edit can be a member of multiple sets, and that is true of many of the edits translated from GenEDITS metafiles.

The concept of a set is used in SEER*DMS to support the way that edits are organized in GenEDITS. Sets cannot be created in SEER*DMS and a registry edit can only belong to the set called All Registry Edits.  See the Edit Sets section of this chapter for more information.

### Conditions

A condition is a Boolean expression that can be associated with an edit. In a particular patient set or record, an edit will only run if all of the edit's conditions return true. Each condition is a small Groovy script and, typically, should be related to a single attribute. You may then associate a set of conditions with an edit. For example, if you were writing an edit for a Treatment Procedure field then you might have conditions related to the TX Proc and the CTC:

- Treatment is not deleted
- CTC is not deleted
- CTC is SEER rpt

SEER*DMS will set default conditions when Data Type is selected. Conditions are set to the defaults defined for the edit's set, if those values are appropriate for the Data Type.

The syntax for the conditions are shown in the Conditions tab at the bottom of the screen.

### GenEDITS Name

In the GenEDITS software, edits have a name but not an ID. SEER*DMS requires a unique ID that meets certain restrictions on length and the type of characters. An ID is assigned to each edit that is translated from GenEDITS. The GenEDITS name field is not used in SEER*DMS, but it is stored as a reference. It is not necessary to enter a GenEDITS Name for a registry edit.

### Source

The source of an edit is determined by the original location of the edit's Groovy code. Registry edits are defined in the SEER*DMS interface by registry staff. The source of these edits is the registry name.

Other edits are loaded into SEER*DMS from XML files. A separate source is defined for each XML file:

- SEER edits are loaded from XML files that are also used in the SEER*Edits software.
- SEER Extended edits were defined by NCI and the SEER*DMS CCB. These edits validate fields that are not required by SEER, but are used by most SEER*DMS registries. IMS staff maintain the SEER Extended edits.
- Translated Edits - 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. The SEER*DMS development team wrote a compiler that translates GenEDITS metafiles into the Groovy scripting language. The source in SEER*DMS is based on the metafile name. The sets for edits from GenEDITS are "Translated from <filename>" for edits from each file.

### Depends On, Depended on by

A dependency is designed to prevent an edit from executing if another edit fails. Typically, a dependency is used to ensure that a valid value is provided for a required field. For example, several SEER edits require age at diagnosis and those edits are not executed if the "age_at_diagnosis" edit fails.

Dependencies are not recommended for registry edits. They are supported in SEER*DMS because they are used in SEER edits, but may be phased out in a future version. Dependencies can make it difficult to determine what is preventing an edit from working; and a user may deactivate one edit and not realize that other edits depend on that edit.

- An edit will not run if it depends on an edit that fails or is inactive.
- An edit that depends on this rule will not run if this edit fails or is inactive.

### 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.

### Definition

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 are 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.

## Edits Manager Tabs

The following tabs are available in the bottom panel of the Edit Editor. Click **Hide/Show Tabs to toggle the view.**

### 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.

### Contexts

In SEER*DMS, contexts are used to define arrays, hash tables, and functions used by the edits. For more information, see the Contexts section of this chapter.

### 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 in the tab at the bottom of the screen.

To find an example of an edit using this function, search for "getEarliestAdmissionForCtc" in the Functions tab. To call this function, you would use this statement:

```
def earliest = Functions.getEarliestAdmissionForCtc(ctc)
```

### 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.

### Properties

Use the Properties tab to search for field documentation and the appropriate syntax for referencing a field in an edit. You must use the Java Path to reference a field in the source code of any edit.

### 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.

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.

## Edit Sets

A set is simply a group of edits. An edit will not be executed unless it is a member of at least one active edit set. An edit can be a member of multiple sets. To view the Validator Set Manager, select **System > Edits Manager > Sets.** All active and inactive Edit Sets will be displayed, along with the source and description.

Some sets are required by SEER*DMS. For example, database constraint edits are required. Required sets are read only and cannot be deactivated.

If the Active flag associated with an Edit Set is unchecked, any edit in the set will not run unless it is in another set that is active.

Default conditions are assigned, by default, when creating a new edit that is in the set. For example, a default condition is automatically assigned when you create a new registry edit. It appears in the editor when you select a Data Type. The default is only assigned if the condition is appropriate for the data type. For example, a condition based on patient set fields will not be used as the default for record edits. For registry edits, you should select a default that is appropriate for most of the edits that you will write.

Default conditions are also assigned to new edits created when GenEDITS metafiles are added to SEER*DMS. Metafile edits are assigned to the same sets as they use in GenEDITS.

Edits are derived from one of the following sources:

1. SEER Edits validate fields submitted to SEER and are defined by 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.

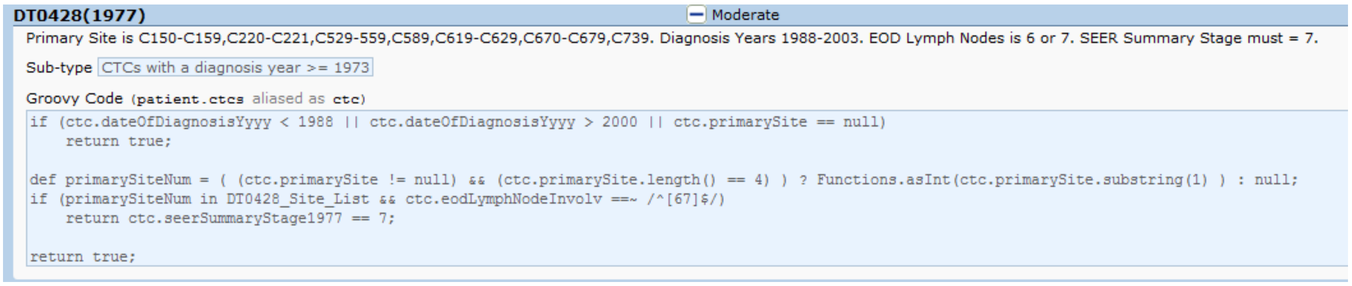
## Contexts

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 registry 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;
```

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.

## 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. This task executes SEER vSE16-016-01 and local patient set edits for a specified cohort.

After the edits are executed, the following polishers are run:

- CTC Visual Editing Flag Reset (Patient Set)
- Date Tumor Record Available (Patient Set)

*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 DX Year** and **End DX 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. You may execute up to six 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).
6. You may provide a comma or space-separated list of Patient Set display **IDs**; only those Patient Sets will be edited.
7. Enter the number of threads to use; must be between 1 and 20
8. 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).
9. 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 involve.

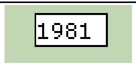
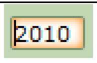
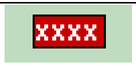
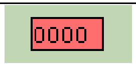
## System Reports Related to Edits

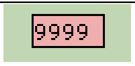
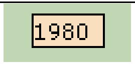
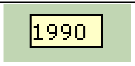
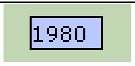
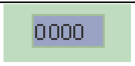
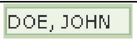
| Report ID | Title | Description |
|---|---|---|
| RPT-064A | Frequency of Edit Errors in Patient Set Data | Use this report to evaluate the error levels in the Patient Set data |
| RPT-064B | Patient Sets with Edit Errors | A listing of Patient Sets with a failure related to a particular edit or a 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. |
| EXT-07 | Edit Rules Extract | Executes the edit rules extraction. |

## 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.

| Example | Description |
|---|---|
|  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). |