SEER*Abs v2.3

System Administration Reference

April 17, 2012

Table of Contents

Section 1: Using SEER*Abs in a Cancer Registry

SEER*Abs was designed using an extensible architecture so that it can be used by any cancer registry. The screen layouts, search tools, extract files, integrated edits, and synchronization module can be configured to meet the needs of the registry. The synchronization module can be configured to load reference data from external files or directly from the registry's main database, regardless of the registry's database platform.

The registry's information technology (IT) staff are responsible for configuring, deploying, and maintaining SEER*Abs. The technical skills required by the system administrator are described below.

Configuring SEER*Abs – Basic Programming Skills Required
In order to create custom layouts or configure other SEER*Abs components, the SEER*Abs system administrator must be an IT professional who has the ability to modify small programs (scripts) and XML configuration files. They must have the ability to write and optimize scripts using the Groovy scripting language. Groovy is a scripting language for the Java platform and uses syntax that is very similar to Java. Online tutorials and references are available at http://groovy.codehaus.org.

Management of Laptop Installations
The registry's SEER*Abs administrator must oversee the deployment of the registry's version of SEER*Abs to the abstractors' workstations. This involves configuring the SEER*Abs Installer software for the registry. Basic computing skills are required including a working knowledge of file and folder structures in the PC environment.

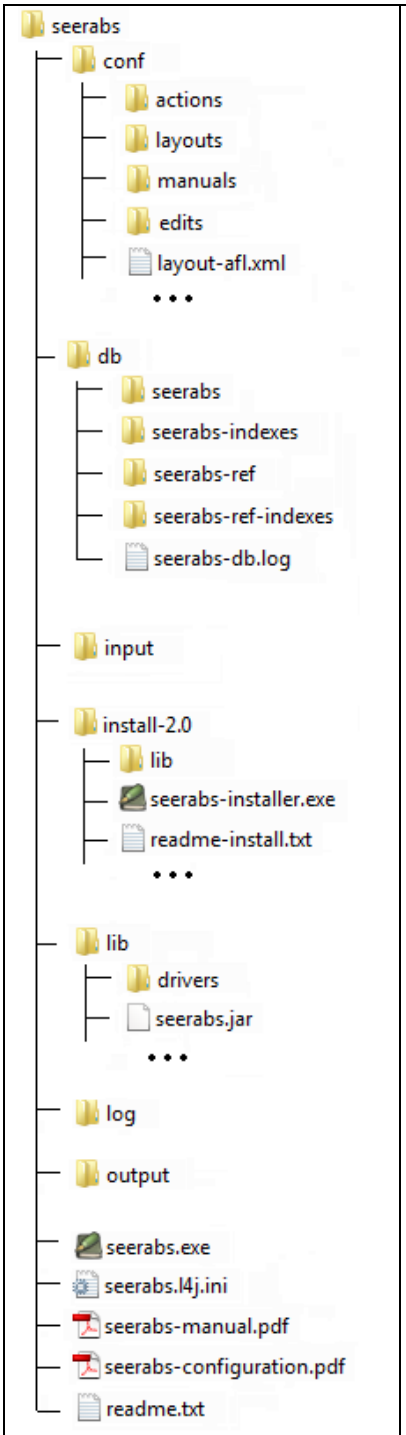
Data Management
The registry's SEER*Abs administrator will oversee the synchronization of data from SEER*Abs with the registry's data management system (DMS). They will need to modify export scripts in SEER*Abs to ensure that the data collected by the abstractors are exported to files that can be imported into the registry's DMS. They may need to pull data from the registry's DMS to populate lookups, facility lists, physician lists, and reference data (the amount of data pulled from the registry's DMS varies and is based on registry customization of SEER*Abs). These tasks require a basic understanding of the registry's data management system.

Getting Started

Spend at least one day seriously reviewing the SEER*Abs Demo Version to understand the features and test drive SEER*Abs before configuring the system for your registry.

1. Perform an Admin Installation of SEER*Abs as described in the SEER*Abs Installation Guide.
2. Review the folder structure for the admin installation of SEER*Abs (shown below).
3. Login to SEER*Abs and explore the layouts and scripts provided in the distribution version of SEER*Abs and/or the demonstration version. Review the SEER*Abs Users Manual to gain an understanding of the application from the abstractor's perspective.
4. Review all chapters in this reference to understand the capabilities of the software. Experiment by making changes to layouts and scripts. Create records within the system to see how your changes look and work.

5. Once you have a working knowledge of the system, define the layouts and customize the actions for your registry.



- **seerabs** – main application folder. SEER*Abs must have read/write access to that folder. SEER*Abs will auto-create sub-folders and configuration files within the folder.
- **conf** – configuration folder. The files in this folder define the screen layouts, actions, import and export routines, etc. The files in the conf folder are required by SEER*Abs and cannot be renamed. Although the conf files are text files, it is recommended that changes be made through the SEER*Abs configuration editor rather than an external editor.
- **db –** database folder. The db/seerabs older contains the main database containing data created within SEER*Abs. The db/seerabs-ref folder contains reference data to use when abstracting. See the SEER*Abs Database section of this manual for more information.
- **input** – default location of files imported into SEER*Abs. This is only a default, import files can be loaded from other locations.
- **install-2.0** – SEER*Abs installer files. The folder name indicates the version. Refer to the SEER*Abs Installation Guide for more information.
- **lib** – library folder containing JAR files required by SEER*Abs.
- **lib/drivers** – folder of database drivers. SEER*Abs auto-loads the drivers in this folder. Drivers are only required if SEER*Abs is configured to synchronize directly with the registry's main database. The default installation includes PostgreSQL JDBC drivers.
- **log –** location of text files containing log messages. All error messages are written to the log including errors caused by registry-maintained scripts and layouts. If an abstractor reports a problem in using SEER*Abs, the system administrator should review the log file within the abstractor's installation.
- **output** – default location of files extracted from SEER*Abs. This is only a default, scripts control where the data are written.

Section 2: SEER*Abs Databases

SEER*Abs databases are implemented as Apache Derby databases (http://db.apache.org/derby). There are two databases in SEER*Abs:

Main Database

The main database is a read/write database containing records created within SEER*Abs, user account information, and AFLs. AFLs may be imported from files or loaded from the registry's database, they are included in the main database because they can be modified by the abstractor and exported.

Indexes for the main database are stored in the db\seerabs-indexes folder. If this folder does not exist, the indexes will be auto-generated when SEER*Abs is started.

Reference Database

The reference database is a read-only database of patient data imported from the registry's main database. Registry configuration settings determine the amount and type of data that are included in reference data. These data may include consolidated patient data from the registry database, pathology records, or other types of records. In addition, lookups, physician lists, and facility lists are also stored in the reference database.

The indexes for the reference database are stored in the db\seerabs-ref-indexes folder.

Reference databases from one installation can be imported into another installation using the Import Reference Database item on the File menu. This complete replaces the reference database in the target installation.

SEER*Abs Data Types

SEER*Abs handles data using the concept of "entity". An entity is an instance of a particular type of data, for example an abstract record, or an AFL. Every entity has a type associated with it; those types cannot be customized and are described in the following table.

Entity Type	Type name to use in Scripts	Database	Description
AFL	AFL	main	Abstract Facility Lead. This entity provides a mechanism for assigning and tracking a request for an abstract. AFLs are imported into SEER*Abs and can be used as a list of "things to do" for the abstractor. The fields used in an AFL can be defined by the registry.
Record	RECORD	main	Records created in SEER*Abs. SEER*Abs can used to create abstract, casefinding, or registry-defined types of records. Data cannot be imported into SEER*Abs record entities.

Entity Type	Type name to use in Scripts	Database	Description
User	USER	main	SEER*Abs user accounts. SEER*Abs supports a single administrative user account (username = admin) and multiple abstractor accounts. The admin user account is created during the initial installation on the administrator's computer. Users cannot be imported; they need to be created within the application.
Facility	FACILITY	reference	The facilities associated with the registry can be defined in facility entities. These may include hospitals, labs, etc. A default facility can be entered during the login process. Facility lists are imported and cannot be modified in SEER*Abs.
Physician	PHYSICIAN	reference	The physician entity is used to store the physicians associated with the registry. This allows the user to select the physician from a lookup while entering record data. The physician list is imported and cannot be modified in SEER*Abs.
Reference Record	REFERENCE-RECORD	reference	The Reference Record entity can be used to store individual reports (non-consolidated data) from the registry's DBMS. The registry can configure the system to include certain types of records (pathology reports, for example) in the system as a reference for the abstractors. Reference data are imported and cannot be modified in SEER*Abs.
Reference Patient	REFERENCE-PATIENT	reference	The Reference Patient entity is designed to store consolidated patient reference data. These data could be imported from files or loaded from the registry's DBMS. Reference Patient data cannot be created or modified in SEER*Abs.
Lookup	N/A *	reference	Lookup, used to provide a list of valid codes for a field.

** Lookups are handled a little bit differently than the other entity types since they don't have customizable properties. For that reason specific methods have been added for them in the script utility methods (see inline help for script methods).*

The second column provides the string that needs to be used when referencing a particular type in a script (many utility methods require a type as a parameter). The third column shows the database in which the entity is persisted.

Subtypes

While it is true that the types are not customizable, some of them have a subtype which is customizable. It is true for the Record and Reference Record types. Which subtype they support is defined in the main configuration file. For the records, the following property is used:

```
supported.record.subtypes=abstract
```

And for the reference records, the following is used:

```
supported.ref.record.subtypes=naaccr
```

Those lists of subtypes can be modified; there is no restriction on the values of the reference record list, but "abstract" is required in the record list.

Defining a New Record Type

Use the following steps to add a new record type:

1. Edit the "supported.record.subtypes" property in the main configuration file, add a new subtype ID for the record type you would like to create (ID should be kept short, for example "casefinding", "short_hrec", special_study", etc…). Let's assume we are adding support for casefinding, the updated property would look like this:
   ```
   supported.record.subtype=abstract,casefinding
   ```

2. In the same configuration file, add a new property for the prefix:
   ```
   record.prefix.casefinding=CF-
   ```

3. In the same configuration file, optionally update the properties that determine which recod type can be copied from which other record type (if not configured, you won't be able to create other record type from the new one and vice-versa). See the comments in the configuration file for more details on how those rules are defined.

4. In the lookup configuration file, add a label for the new record type:
   ```
   <lookup id="lkup_internal_rec_subtype">
       <entry code="abstract" label="Abstract" />
       <entry code="casefinding" label="Casefinding" />
   </lookup>
   ```

5. Restart the application.

6. In the Data Entry configuration tab, under "Supported Record Types", edit the XML file that determines which fields are displayed in the data entry form and how. Note that the XML file also defines which fields are supported by this record type, so supporting a new field for our new casefinding type is as simple as adding the field in the XML file. The configuration file editor contains a lot of inline help about creating and maintaining the data entry forms.

7. In the Data Entry configuration tab, under "Supported Record Types", edit the XML file that determines how the records for the new type will be extracted. Although other methods are supported in SEER*Abs, creating a data file is by far the most common one. The extract is defined as a Groovy script that takes the output file as a parameter. The default version

queries the database, fetches all the records of that particular type that are ready to be extracted and output them in the file, one by one. The script is fully customizable.

Defining New Properties

Lookups are a special type of entity and are defined in their own configuration file (see *Defining Lookups* section). All the other entity types are defined in a Layout. That layout contains the properties that should be shown on the screen along with some other information (property type, label, lookup, etc…). While it is true that the layout is mainly used to define where the fields should be shown on the screen, it is also used to define which properties are supported for which entity type. An entity can be seen as a map of keys and values. The keys are the field names defined in the layout and the values are the text corresponding to those field names (it can be the text typed by the abstractor in the editor, or the text downloaded through the synchronization module). For efficiency, a key corresponding to an empty (null) value is not saved in the database; that means the absence of a key in a map should be interpreted as the key having an empty value. That also means different entities of the same type will end up with different keys, depending which values are missing. For that reason, there are no database constraints linking the properties to the entity types; saving an entity in the database means saving a generic mapping of keys and values; the database is unaware of which properties the mapping should have depending on the entity layout.

With this design, adding a new property to a given type is as simple as adding a field to the corresponding layout. Once added, the abstractor will be able to provide a value to that field; that value will be persisted in the database and made available to the synchronization scripts to be exported. Any properties can be defined in a layout, but a few of them are used by the application and therefore SEER*Abs needs to be aware of their name. Most of those internal properties can be re-defined in the main configuration in case they conflict with other regular properties.

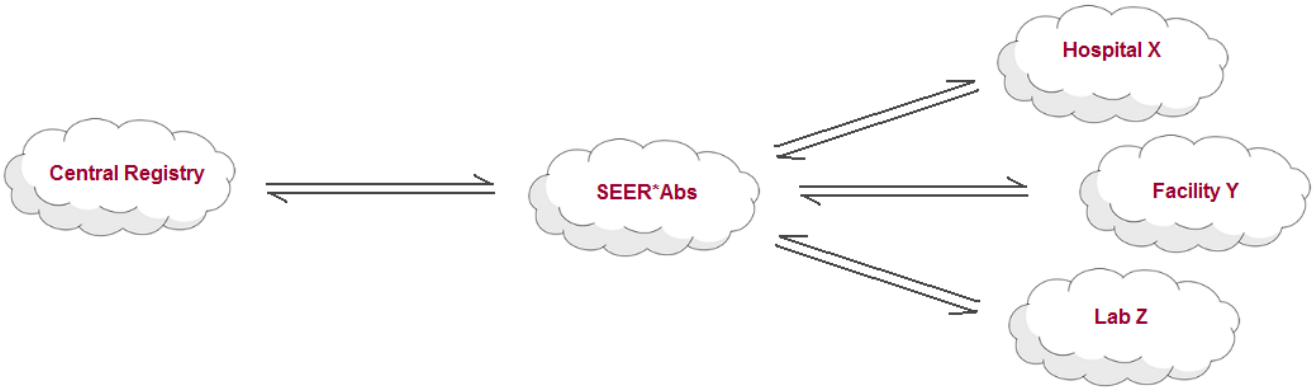
There are a few other properties used internally by SEER*Abs (like a database ID for example) but those should never be referenced by any scripts and therefore are not described here (they usually start with a double underscore).

Having to re-define an internal property in the main configuration should be extremely rare. For example if a new reference record type is added and has to use the property "dateLastModified", the internal property with the same name could be re-defined as "dateLastModifiedSeerabs" to avoid any conflict. But the script downloading that new reference record type could also save that new "dateLastModified" property under a different name and therefore also avoid the conflict. Note that if a property is re-defined, all the data needs to be fixed (for reference data, it means deleting the old data and re-importing it; for the main data it means running an action script that would load all the entities of that type and for each of them remove the old property and re-add the new one).

Because empty values are not saved in the database, different entities of the same type could have different properties saved in the database. For that reason, a script cannot make any assumptions on which properties is supposed to be contained in an entity. This can be annoying when trying to write scripts that reference hundreds of properties. To solve that problem, a utility method is provided; for a given type and subtype, it returns a list of properties as they are defined in the corresponding layout. See the *Script Methods* help menu for more details about utility methods.

Section 3: SEER*Abs Workflow

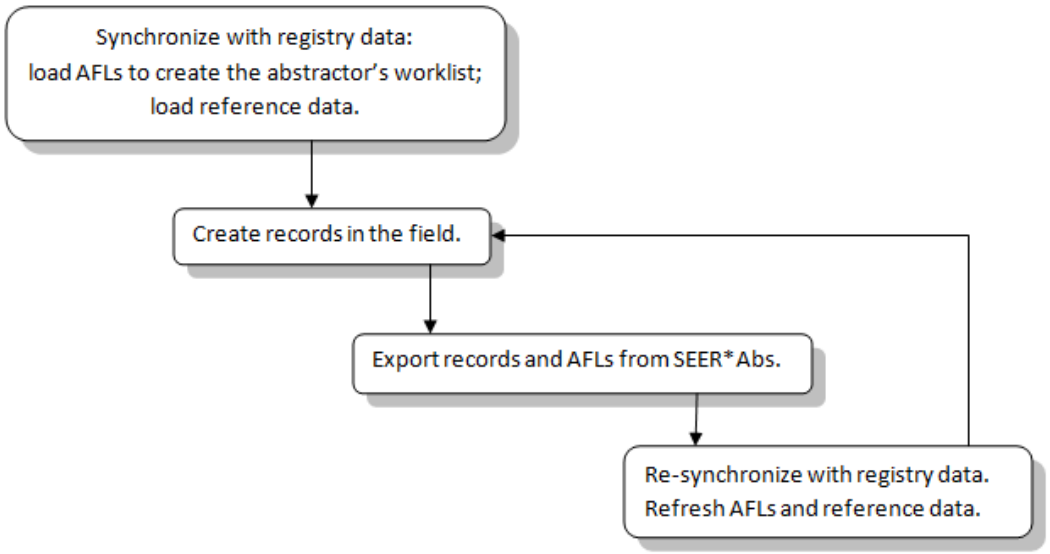
Before configuring SEER*Abs, it is important to decide how it will be used in your registry. As shown in the diagram below, SEER*Abs facilitates the flow of data to and from the abstractors in the field to the registry. Configuration settings determine the amount and type of data made available to abstractors as a reference; and the types of data collected within SEER*Abs.



SEER*Abs is designed as an Abstracting Tool to be used in the field to create abstract records and other types of records.

An important aspect of the workflow is for the Abstractor to be able to organize her work. This is accomplished through the Worklist page which displays listings of Abstract Facility Leads (if they are turned on in the main configuration file), and abstracted records. The worklist can be filtered to show only the outstanding work.

SEER*Abs communicates with the registry through the synchronization module to export the created records and maybe import the reference data. That default workflow is shown in the following figure:



By configuring the synchronization module, a Registry can define how the records and AFLs are exported and how the reference data is imported. By configuring the editor module, a Registry can define how the records are created (what fields, what format, etc...).

Abstractors need to know whether a record has just been created, whether all the work is done for it or whether it has already been exported. A status field is used for that purpose. Only the AFL and the RECORD entity types have that field. The possible values are defined in the lookups *lkup_internal_alf_status* and *lkup_internal_rec_status*. Those lookups cannot be deleted from the configuration but their content can be modified and that is the main mechanism to customize the SEER*Abs workflow.

The following AFL statuses are provided with the default configuration:

Status Code	Status Label	Description
1	NOT PROCESSED	No work has been performed on this AFL.
2	IN PROGRESS	Some work has been performed on this AFL
3	NOT ABSTRACTED	The work on this AFL is done; it has not been abstracted and a reason has been provided.
4	ABSTRACTED	The work on this AFL is done; an abstract has been created.
5	ARCHIVED	This AFL has been exported.

The following record statuses are provided with the default configuration:

Status Code	Status Label	Description
1	IN PROGRESS	Some work has been performed on this record.
2	COMPLETED	No more work needs to be performed on this record.
3	ARCHIVED	This record has been exported.

Note that changing the label of a status has no impact on the workflow and no script needs to be modified in that case. On the other hand, many scripts use the status code to search entities and load them (for example the export script fetches all the records with a status of 'COMPLETED'). Adding, removing or changing the meaning of a status (how it is supposed to be used by the scripts) require to review each script and make sure the way it uses the statuses (if it does use them) is still correct.

SEER*Abs never deletes any record or AFL automatically. A special action is provided (Purge Entities) to delete any entities with a status of 'ARCHIVED'. That action can be run right before exporting records so the ones exported from the previous synchronization session are deleted and

the new ones are marked as ‘ARCHIVED’ after being exported. But the user has to trigger the action manually and exactly when that should happen must be a Registry policy.

Section 4: Configuring SEER*Abs

The SEER*Abs configuration manager can be used to customize all system features including data entry screens for records, the Search page interface, and synchronization scripts. The manager allows you to open files in the SEER*Abs configuration editor by clicking the "Edit" button of the corresponding file.

The configuration manager, as well as the configuration file editor, contains extensive help within the application.

The configuration files fall in one of the following topics:

- **Main Configuration** – a single configuration file containing system parameters and properties. These include system options, defaults, and global variables.
- **Layouts** – separate configuration files defining the screens displayed when you view or modify a record, AFL, patient set, facility, physician, or user account. These include the layouts used to display records created in SEER*Abs and reference patient sets and records.
- **Customized Popups** – layouts for dialogs to prompt the user for information. These can be referenced in any script to request user input.
- **Searches & Filters** – these files define the filters and tables used to display and find data on the Search page, the Worklist, and the User account manager.
- **Scripts** – Groovy scripts run to create extract files, load reference data from external files or directly from a database, and utility scripts used to create AFLs and implement edits.
- **Scripts Shown in the Action Menu** – special Groovy scripts that the user can execute by selecting the item from the "Action" menu.
- **Lookups** – a single configuration file defining lookup tables for data fields. This file contains full definitions for internal lookups and reference information for external lookups.
- **Edits** – source code for edits to test the validity of data fields. Two sets of edits are integrated into SEER*Abs: edits developed by the NCI SEER Program that cover data fields submitted to SEER; and edits developed within SEER*Abs. The SEER*Abs edits may include edits shipped with the software as well as the edits created by your registry staff.
- **Autocomplete Terms** – the list of words and phrases matched against the user's entry when the autocomplete feature is used.
- **Manuals** – the manuals appearing to the abstractor through the help menu. Manuals can be removed, updated or added.

Each of those topics is explained in details in the following sections.

Main Configuration

Global configuration parameters are set in a main configuration file "seerabs.properties". The file can be accessed through the Main Configuration section in the Configuration page.

Review and adjust the global parameters before configuring the layouts and scripts. Changes to some parameters are applied when you save and close the configuration editor. Other changes are not applied until you close and restart SEER*Abs, as prompted. Any changes made via a text editor will only be applied when you restart SEER*Abs.

Every parameter is listed in a table in the Configuration page. Click on a label to go the parameter's definition in the configuration file editor. The main configuration file also contains extensive help for each parameter.

Defining Layouts

XML configuration files define the display screens for record, AFL, patient set, facility, physician, and user account data. There is a separate XML file for each of the following:

- AFL page
- Data Entry screen for each record type defined in the main configuration
- Screens to view reference data:
    - Facility
    - Physician
    - Any record type defined in the main configuration
- User Account page

The layout files should be edited via the SEER*Abs editor to take advantage of the validation, preview, and auto-refresh features. However, the XML files are stored in the *conf* installation folder and can be opened with any text editor.

Instructions on how to update the layouts are provided in the Configuration page.

Layouts can also be used to define User Input dialogs that can be called from Groovy scripts. See the Customized Popups section in the Configuration page for more information.

Configuring Searches & Filters

XML configuration files are used to define screen layouts for the Worklist table, User account manager, the three tabs of the Search page, the facility lookup, and the physician lookup. These configuration files are defined as layouts with two sections: criteria-layout defines the filters and the table-layout defines the table in which the results are displayed. Filters cannot always be defined. The search and filter layout files are listed below.

- **Facility** (search-facility.xml) – the layout of the Facility tab of the Search page.
- **Facility Lookup** (search-facility-lkup.xml) – the layout of the internal lookup for facilities. A single search box is shown in that lookup; for that reason the criteria defined in the configuration file is not used to show different search fields, but instead it is used to know which fields should be searched when the user types text in the unique search box.
- **Patient Data** (search-patient.xml) – the layout of the Patient Data tab of the Search page.

- **Physician** (search-physician.xml) – the layout of the Physician tab of the Search page.
- **Physician Lookup** (search-physician-lkup.xml) – the layout of the internal lookup for physicians. A single search box is shown in that lookup; for that reason the criteria defined in the configuration file is not used to show different search fields, but instead it is used to know which fields should be searched when the user types text in the unique search box.
- **Users** (search-user.xml) – the layout of the Users Account manager. No criteria can be defined for that configuration file.
- **Worklist** (search-worklist.xml) – the layout of the Worklist. The filter in the worklist cannot be customized but it does use a free-text search box. The criteria section defined in this configuration file is used to know which fields should be searched when the user types text in that free-text search box.

The searches in SEER*Abs are implemented using an external library called Lucene (http://lucene.apache.org/java/docs).

Instructions on how to update the search layouts are provided in the Configuration page.

Defining Scripts

Groovy is a scripting language for the Java platform. The Internet has several Groovy references including the Groovy home page at http://groovy.codehaus.org. The official site contains a lot of information, including tutorials for people new to Groovy.

The Configuration page contains many application scripts that can be maintained using the configuration file editor. The editor contains additional information and help on how to write and maintain the scripts.

Scripts can be made available under the Action menu item. Action scripts are very similar to regular scripts; the difference is that they can be added, removed or modified without any consequences in the application; while only the content of a regular script can be modified (for example the extract script for abstract is called *script-extract-record-abstract.groovy*; deleting that file outside of the application will result in a failure during the startup process). Because the action scripts can be added and removed, they have their own directory (conf/scripts/).

Writing action script is not different than writing regular scripts; the scripting language (Groovy, see http://groovy.codehaus.org) is identical. One minor distinction is that the regular scripts usually receive data in their context (for example, the script that runs when a record is saved receives that record in its context so it can be modified by the script) while the action script do not receive any data in their context (since there are triggered by the user selecting them from a menu item).

Once an action script has been defined, it is available in the *Action* menu. The default configuration provided with SEER*Abs contains a single action script called "Purge Entities"; it deletes from the database any AFL or RECORD that have a status of ARCHIVED.

Defining Lookups

Lookup tables provide a list of valid values for a field. Typically, this is a list of codes and a user-friendly description of the code. When a lookup is associated with a field in a layout, a light bulb is

displayed next to the field. The lookup table is displayed when the user clicks the light bulb, they may then select a value from the list.

There are five types of lookups: standard, facility, physician, collaborative stage, and site specific surgery. The standard lookup is a mapping of unique codes to labels. Standard lookups are defined in a single configuration file (lookups.xml). The definition may consist of the full mapping or may indicate that the mapping is to be imported using one of the synchronization scripts.

The other four types of lookups are more complex and cannot be defined in XML. The facility and physician lookups can be customized. The site-specific surgery and collaborative stage lookups cannot be modified or customized.

Instructions on how to maintain the lookups are provided in the Data Entry section of the configuration page under the Lookups tab.

Defining Edits

Computerized edits are integrated into SEER*Abs to test the validity of data. In SEER*Abs, the edits are executed on records created in SEER*Abs, they are not executed on reference data. The following sets of edits are available in SEER*Abs:

- Internal system edits enforce data type constraints in layouts. The system edits cannot be modified, they are defined in the Layout XML file (for example defining a field as numeric will trigger a system edit if the abstractor tries to enter a non-numeric value).
- SEER*Abs edits are defined and maintained by registry staff. Samples are provided in the configuration file shipped with the application. SEER*Abs edits apply to any record types created in the application, but it is possible to restrict them to some particular types from within the edit file itself.
- SEER edits cover fields submitted to SEER and represent the edits implemented in the SEER*Edits software. The SEER edits are defined in XML files provided and maintained by the SEER*Edits development team. The SEER edits configuration file cannot be modified. By default the SEER edits are not loaded in SEER*Abs but that behavior can be changed through a configuration variable in the main configuration file. If loaded, the SEER edits are applied to any record type having an ID starting with "abstract" (so in the default configuration, they are applied only to the Abstract records).
- NPCR edits are edits translated from the NPCR set in the NAACCR metafile. Those edits are maintained by IMS and cannot be modified; they are automatically updated by the installer when a new version of the metafile becomes available. By default the NPCR edits are not loaded in SEER*Abs but that behavior can be changed through a configuration variable in the main configuration file. If loaded, the NPCR edits are applied to any record type having an ID starting with "abstract" (so in the default configuration, they are applied only to the Abstract records).

In addition, other sets of edits can also be loaded; refer to the help on the Edits tab of the Data Entry section in the Configuration page.

SEER*Abs edits are implemented in Groovy, the scripting language for the Java platform that is also used for SEER*Abs scripts. Edits uses a small subset of the Groovy syntax. A working

knowledge of regular expressions and Groovy logic statements are needed to maintain edits in SEER*Abs. To define a new edit, it is recommended that you copy-and-paste the code from an existing edit and use that code 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.
- Use the Groovy code of a similar edit as a template.
- A context is a Java naming system. 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. Contexts are defined within the XML context tag. Many examples are provided in the SEER edits XML file. Your Groovy code may include references to contexts that you define and the contexts defined for the SEER edits.

Defining Autocompletion Word Lists

While entering text in a field, an abstractor may press Ctrl+Space to use the autocomplete feature (refer to the SEER*Abs Users Manual for more information). Autcomplete is available for string and unlimited-string fields in record and AFL layouts.

SEER*Abs supports multiple sets of autocomplete terms. Separate lists may be designated for different fields, for example, there may be one list for histology and another for primary site. Or terms from all lists may be made available in a field, for example, all terms are typically made available when editing large text fields. When a field does not define any autocomplete list in the layout definition, it will automatically use the "default" list. For that reason, there must always be a list with a name "default" define in the autocompletion word lists.

Instructions on how to maintain the autocompletion word lists are provided in the Data Entry section of the configuration page under the Autocompletion tab.

Managing Coding Manuals

SEER*Abs is shipped with several common coding manuals, but those can be removed and others can be added through the Coding Manuals section in the configuration page.

All manuals are kept in the "conf/manuals" folder; the manuals are typically PDF files, but can be on any type as long as the laptop can handle the file type.

Instructions on how to add, remove or modify manuals are provided in the Coding manuals section of the configuration page.

Section 5: Managing User Accounts

SEER*Abs supports a single administrative user account (username = admin) and multiple abstractor accounts. The admin user account is created during the initial installation on the administrator's computer. The system administrator configures SEER*Abs and creates a registry-specific installation file. Registry managers and the SEER*Abs system administrator must define a protocol for maintaining abstractor accounts.

- A single abstractor account may be created as SEER*Abs is installed on each workstation. The abstractor using that computer would then complete the installation by defining a password known only to them.
- Alternatively, the system administrator may create accounts for all abstractors during the initial configuration. A protected list of unique passwords would be created. Accounts for all abstractors would then be installed on all workstations.

There is no method for synchronizing the user accounts on multiple installations of SEER*Abs. Once the system is deployed, you will need to add and remove users from each installation; or modify a central version and re-install the software.

To add, modify, or delete user accounts:

1. Login as the administrator.
2. Open the Users Manager.
3. To add a new account, click **Add User**.
    a. Provide a username and the initial password for the new user. The password must contain at least 1 lower-case letter, 1 upper-case letter, and either a digit or a special character. This is the password that the user will enter the first time they login. They will then be prompted to specify a password known only to them.
    b. You may enter values for the user-defined fields (optional). These fields are defined in the User layout configuration files.
    c. Click **Save**.
4. To delete a user account, click the X icon associated with the user account. The admin account cannot be deleted.
5. To define a new password for an account:
    a. Double-click the username or click the edit icon.
    b. Click **Change Pswd**
    c. Enter the new **Password**.
    d. Verify the new password by re-entering it into the **Repeat Password** field.
6. To set or change the values of registry fields:

a. Double-click the username or click the edit icon.

b. Enter new values for the registry defined fields.

Section 6: Data Security

The SEER*Abs software is a data collection tool specifically designed for confidential medical data. Security controls are built-in to the software as described below. However, these features do not ensure full security and must be considered as only part of the registry's security plan. SEER*Abs must be used on a workstation that is physically secure and/or fully encrypted.

SEER*Abs provides the following mechanism to secure the data:

- Strong passwords:
    1. Password must be 8 characters long
    2. Password must contain 3 of the 4 groups: upper, lower, digit and special characters
    3. Password must not contain three consecutive identical characters
    4. Password must not share a sequence of 6 or more characters with the previous password
    5. Password must not contain the username
    6. Password must not be the same as the 24 previous passwords

  The Admin user is a super-user and therefore only rules 1 to 3 apply to that user.

- Password expiration: by default, password will expire after 60 days. Although it is not recommended to do so, the expiration time can be increased in the main configuration file; it is also possible to disable the expiration entirely.

- Database encryptions: Derby is a file-based Java database stored in binary format. The data are encrypted by Derby using the Data Encryption Standard (DES) 56 bits algorithm. That algorithm is tied to a key that is required to make any interaction with the databases. Without the key, external programs (implemented in Java or other languages) will be unable to create a connection to the database.

When evaluating the data confidentiality aspects in SEER*Abs, keep the following in mind:

- The sample extract scripts distributed with SEER*Abs do not create encrypted files. The extract scripts are maintained by registry staff and can be modified to create encrypted files.

- SEER*Abs extracts are written to a directory that is selected by the user or defined by registry IT in the configuration files. It is the registry's and the user's responsibility to write the file to a secure location and to securely transfer the data to the registry.

- Data may be transferred directly between SEER*Abs and the registry database. The data transfer is not encrypted. The synchronization needs to be done behind the registry's firewall or through a secured VPN.

- The database encryption is not unbreakable. It is the responsibility of the registry to ensure physical security of the workstation. If used on a laptop in the field, the entire hard-drive of the laptop must be encrypted.

Resources:

- NIST Computer Security Resource Center (http://csrc.nist.gov)