EVERGREEN IN ACTION

Published : 2014-05-15
License : None

ABOUT THIS BOOK

1. ACKNOWLEDGEMENTS

Thanks, first and foremost, to the Google Open Source Programs Office for providing us with food, shelter, transportation, and internet access during the Doc Sprint. Carol Smith's attention to details made it possible for us to concentrate on our task at hand.

Allen Gunn from AspirationTech shook us up and put us back together as a determined team with his energetic facilitation efforts. Adam Hyde of FLOSS Manuals helped guide us through what would otherwise have been a grueling process with his gentle encouragement and enthusiasm.

Thanks to Bradley Kuhn of the Software Freedom Conservancy for coordinating the financial details for the Evergreen project's participation in the Doc Sprint.

We would especially like to acknowledge the assistance of members of the Evergreen community who stepped up to answer our questions, provide sage advice and unstinting support. Without them, and the many others who contribute to Evergreen in many different ways, we would not see the high level of progress that our community continues to show.

Ben Shum and Jason Stephenson graciously agreed to let us reuse parts of their circulation and holds presentation for examples and screenshots, without which the corresponding chapters of this book would have been impoverished. Similarly, we were pleased to draw upon some of the rich material that the Evergreen Documentation Interest Group has written. Thanks to all of you for your exemplary spirit of sharing!

Finally, we cannot thank our families enough for enabling us to participate in this project. Your sacrifice has enabled us to create this manual.

2. OVERVIEW OF DAILY AND ONGOING OPERATION

The Evergreen integrated library system is a large and complex set of interacting pieces of software. At its heart, though, it fulfills two main functions:

- *inventory management*; keeping track of what the library owns, where it is, and even ordering new materials
- *patron management*; controlling who has access to the materials and their current status

As someone responsible for the whole system, your tasks include reflecting library policies in the system and overseeing the welfare of the system's software and data. On any given day, you and members of your team may find yourselves implementing policies in Evergreen, importing data, tracking budgetary information, customizing interfaces, altering search functionality, setting up notifications, preparing for disaster recovery, updating the Evergreen software and keeping the system secure.

Any one of these tasks can be daunting to both new and experienced Evergreen administrators. The authors are also Evergreen administrators, and we also found a number of these tasks daunting individually. By working together, we were able to pool our experiences--so this manual is just as much for us as it is for you.

WHAT THIS MANUAL DOES FOR YOU

This manual is designed to take you through a number of critical tasks that will face you and to make your life easier while you do them. We assume that you have a working installation of Evergreen and that you are ready to tackle the next steps.

The sections are grouped broadly by function:

- tasks you have to perform when your installation is up and running, before it is ready for patrons and staff to use;
- controlling who has privileges to use your system, at which locations;
- providing the optimal experience for your patrons, with a focus on your catalog and communication;
- preparing for (and recovering from) disasters that may affect Evergreen's data;
- keeping your installation of Evergreen up to date;
- getting help when you need it, and participating in the Evergreen community.

The material is presented roughly in the ideal order in which you would address them. Decisions that you make when you initially load data into and configure your system can have significant consequences when you subsequently define the rules that control how patrons can borrow materials. As an administrator, however, life is not ideal, and you may find yourself called on to handle many different things at once. Consequently, you can use this book either as a straight-through read, or just jump to the sections that stand out most for you right now.

INITIAL SET UP

3. DESCRIBING YOUR ORGANIZATION

Your Evergreen system is almost ready to go. You'll need to add each of the libraries that will be using your Evergreen system. If you're doing this for a consortium, you'll have to add your consortium as a whole, and all the libraries and branches that are members of the consortium. In this chapter, we'll talk about how to get the Evergreen system to see all your libraries, how to set each one up, and how to edit all the details of each one.

ORGANIZATION UNIT TYPES

The term *Organization Unit Types* refers to levels in the hierarchy of your library system(s). Examples could include: All-Encompassing Consortium, Library System, Branch, Bookmobile, Sub-Branch, etc.

You can add or remove organizational unit types, and rename them as needed to match the organizational hierarchy that matches the libraries using your installation of Evergreen. Organizational unit types should never have proper names since they are only generic types.

When working with configuration, settings, and permissions, it is very important to be careful of the Organization Unit **Context Location** - this is the organizational unit to which the configuration settings are being applied. If, for example, a setting is applied at the Consortium context location, all child units will inherit that setting. If a specific branch location is selected, only that branch and its child units will have the setting applied. The levels of the hierarchy to which settings can be applied are often referred to in terms of "depth" in various configuration interfaces. In a typical hierarchy, the consortium has a depth of 0, the system is 1, the branch is 2, and any bookmobiles or sub-branches is 3.

Create and edit Organization Unit Types

1. Open **Admin > Server Administration > Organization Types**.
2. In the left panel, expand the Organization Unit Types hierarchy.
3. Click on a organization type to edit the existing type or to add a new organization unit.
4. A form opens in the right panel, displaying the data for the selected organization unit.
5. Edit the fields as required and click **Save**.

To create a new dependent organization unit, click **New Child**. The new child organization unit will appear in the left panel list below the parent. Highlight the new unit and edit the data as needed, click **Save**

ORGANIZATIONAL UNITS

Organizational Units are the specific instances of the *organization unit types* that make up your library's hierarchy. These will have distinctive proper names such as Main Street Branch or Townsville Campus.

Remove or edit default Organizational Units

After installing the Evergreen software, the default CONS, SYS1, BR1, etc., organizational units remain. These must be removed or edited to reflect actual library entities.

Create and edit Organizational Units

1. Open **Admin > Server Administration > Organizational Units**.
2. In the left panel, expand the the Organizational Units hierarchy, select a unit.
3. A form opens in the right panel, displaying the data for the selected organizational unit.
4. To edit the existing, default organizational unit, enter system or library specific data in the form; complete all three tabs: Main Settings, Hours of Operation, Addresses.
5. Click **Save**.

To create a new dependent organizational unit, click **New Child**. The new child will appear in the hierarchy list below the parent unit. Click on the new unit and edit the data, click **Save**

Organizational Unit data

- The **Addresses** tab allows you to enter library contact information. Library Phone number, email address, and addresses are used in patron email notifications, hold slips, and transit slips. The Library address tab is broken out into four address types: Physical Address, Holds Address, Mailing Address, ILL Address.
- The **Hours of Operation** tab is where you enter regular, weekly hours. Holiday and other closures are set in the **Closed Dates Editor**. Hours of operation and closed dates impact due dates and fine accrual.

Set closed dates using the Closed Dates Editor

1. Open **Admin > Local Administration > Closed Dates Editor**.
2. Select type of closure: typically Single Day or Multiple Day.
3. Click the Calendar gadget to select the All Day date or starting and ending dates.
4. Enter a Reason for closure (optional).
5. Click **Apply** to all of my libraries if your organizational unit has children units that will also be closed.
6. Click **Save**.

Now that your organizational structure is established, you can begin configuring permissions for the staff users of your Evergreen system.

4. DESCRIBING YOUR PEOPLE

Many different members of your staff will use your Evergreen system to perform the wide variety of tasks required of the library.

When the Evergreen installation was completed, a number of permission groups should have been automatically created. These permission groups are:

- Users

- Patrons
- Staff
- Catalogers
- Circulators
- Acquisitions
- Acquisitions Administrator
- Cataloging Administrator
- Circulation Administrator
- Local Administrator
- Serials
- System Administrator
- Global Administrator
- Data Review
- Volunteers

Each of these permission groups has a different set of permissions connected to them that allow them to do different things with the Evergreen system. Some of the permissions are the same between groups; some are different. These permissions are typically tied to one or more *working location* (sometimes referred to as a *working organizational unit* or *work OU*) which affects where a particular user can exercise the permissions they have been granted.

SETTING THE STAFF USER'S WORKING LOCATION

To grant a working location to a staff user in the staff client:

1. Search for the patron. Select **Search > Search for Patrons** from the top menu.
2. When you retrieve the correct patron record, select **Other > User Permission Editor** from the upper right corner. The permissions associated with this account appear in the right side of the client, with the **Working Location** list at the top of the screen.
3. The **Working Location** list displays the Organizational Units in your consortium. Select the check box for each Organization Unit where this user needs working permissions. Clear any other check boxes for Organization Units where the user no longer requires working permissions.
4. Scroll all the way to the bottom of the page and click **Save**. This user account is now ready to be used at your library.

As you scroll down the page you will come to the **Permissions** list. These are the permissions that are given through the **Permission Group** that you assigned to this user. Depending on your own permissions, you may also have the ability to grant individual permissions directly to this user.

COMPARING APPROACHES FOR MANAGING

PERMISSIONS

The Evergreen community uses two different approaches to deal with managing permissions for users:

- **Staff Client**
  Evergreen libraries that are most comfortable using the staff client tend to manage permissions by creating different profiles for each type of user. When you create a new user, the profile you assign to the user determines their basic set of permissions. This approach requires many permission groups that contain overlapping sets of permissions: for example, you might need to create a *Student Circulator* group and a *Student Cataloger* group. Then if a new employee needs to perform both of these roles, you need to create a third *Student Cataloger / Circulator* group representing the set of all of the permissions of the first two groups.

  The advantage to this approach is that you can maintain the permissions entirely within the staff client; a drawback to this approach is that it can be challenging to remember to add a new permission to all of the groups. Another drawback of this approach is that the user profile is also used to determine circulation and hold rules, so the complexity of your circulation and hold rules might increase significantly.
- **Database Access**
  Evergreen libraries that are comfortable manipulating the database directly tend to manage permissions by creating permission groups that reflect discrete roles within a library. At the database level, you can make a user belong to many different permission groups, and that can simplify your permission management efforts. For example, if you create a *Student Circulator* group and a *Student Cataloger* group, and a new employee needs to perform both of these roles, you can simply assign them to both of the groups; you do not need to create an entirely new permission group in this case. An advantage of this approach is that the user profile can represent only the user's borrowing category and requires only the basic *Patrons* permissions, which can simplify your circulation and hold rules.

Permissions and profiles are not carved in stone. As the system administrator, you can change them as needed. You may set and alter the permissions for each permission group in line with what your library, or possibly your consortium, defines as the appropriate needs for each function in the library.

MANAGING PERMISSIONS IN THE STAFF CLIENT

In this section, we'll show you in the staff client:

- where to find the available permissions
- where to find the existing permission groups
- how to see the permissions associated with each group
- how to add or remove permissions from a group

We also provide an appendix with a listing of suggested minimum permissions for some essential groups. You can compare the existing permissions with these suggested permissions and, if any are missing, you will know how to add them.

Where to find existing permissions and what they mean

In the staff client, in the upper right corner of the screen, click on **Admin > Server Administration** > **Permissions**.

The list of available permissions will appear on screen and you can scroll down through them to see permissions that are already available in your default installation of Evergreen.

There are over 500 permissions in the permission list. They appear in two columns: **Code** and **Description**. Code is the name of the permission as it appear in the Evergreen database. Description is a brief note on what the permission allows. All of the most common permissions have easily understandable descriptions.

Where to find existing Permission Groups

In the staff client, in the upper right corner of the screen, navigate to **Admin > Server Administration > Permission Groups**.

Two panes will open on your screen. The left pane provides a tree view of existing Permission Groups. The right pane contains two tabs: Group Configuration and Group Permissions.

In the left pane, you will find a listing of the existing Permission Groups which were installed by default. Click on the **+** sign next to any folder to expand the tree and see the groups underneath it. You should see the Permission Groups that were listed at the beginning of this chapter. If you do not and you need them, you will have to create them.

Adding or removing permissions from a Permission Group

First, we will remove a permission from the Staff group.

1. From the list of Permission Groups, click on **Staff**.
2. In the right pane, click on the **Group Permissions** tab. You will now see a list of permissions that this group has.
3. From the list, choose **CREATE_CONTAINER**. This will now be highlighted.
4. Click the **Delete Selected** button. CREATE_CONTAINER will be deleted from the list. The system will not ask for a confirmation. If you delete something by accident, you will have to add it back.
5. Click the **Save Changes** button.

You can select a group of individual items by holding down the *Ctrl key* and clicking on them. You can select a list of items by clicking on the first item, holding down the *Shift key*, and clicking on the last item in the list that you want to select.

Now, we will add the permission we just removed back to the Staff group.

1. From the list of Permission Groups, click on **Staff**.
2. In the right pane, click on the **Group Permissions** tab.
3. Click on the **New Mapping** button. The permission mapping dialog box will appear.
4. From the Permission drop down list, choose **CREATE_CONTAINER**.
5. From the Depth drop down list, choose **Consortium**.
6. Click the checkbox for **Grantable**.
7. Click the **Add Mapping** button. The new permission will now appear in the Group Permissions window.
8. Click the **Save Changes** button.

If you have saved your changes and you don't see them, you may have to click the Reload button in the upper left side of the staff client screen.

MANAGING ROLE-BASED PERMISSION GROUPS IN THE DATABASE

While the ability to assign a user to multiple permission groups has existed in Evergreen for years, a staff client interface is not currently available to facilitate the work of the Evergreen administrator. However, if you or members of your team are comfortable working directly with the Evergreen database, you can use this approach to separate the borrowing profile of your users from the permissions that you grant to staff, while minimizing the amount of overlapping permissions that you need to manage for a set of permission groups that would otherwise multiply exponentially to represent all possible combinations of staff roles.

In the following example, we create three new groups:

- a *Student* group used to determine borrowing privileges
- a *Student Cataloger* group representing a limited set of cataloging permissions appropriate for students
- a *Student Circulator* group representing a limited set of circulation permissions appropriate for students

Then we add three new users to our system: one who needs to perform some cataloging duties as a student; one who needs perform some circulation duties as a student; and one who needs to perform both cataloging and circulation duties. This section demonstrates how to add these permissions to the users at the database level.

To create the Student group, add a new row to the *permission.grp_tree* table as a child of the *Patrons* group:

```
INSERT INTO permission.grp_tree (name, parent, usergroup, description,
application_perm)
SELECT 'Students', pgt.id, TRUE, 'Student borrowers',
'group_application.user.patron.student'
FROM permission.grp_tree pgt
 WHERE name = 'Patrons';
```

To create the Student Cataloger group, add a new row to the *permission.grp_tree* table as a child of the *Staff* group:

```
INSERT INTO permission.grp_tree (name, parent, usergroup, description,
application_perm)
SELECT 'Student Catalogers', pgt.id, TRUE, 'Student catalogers',
'group_application.user.staff.student_cataloger'
FROM permission.grp_tree pgt
WHERE name = 'Staff';
```

To create the Student Circulator group, add a new row to the *permission.grp_tree* table as a child of the *Staff* group:

```
INSERT INTO permission.grp_tree (name, parent, usergroup, description,
application_perm)
SELECT 'Student Circulators', pgt.id, TRUE, 'Student circulators',
'group_application.user.staff.student_circulator'
FROM permission.grp_tree pgt
WHERE name = 'Staff';
```

We want to give the Student Catalogers group the ability to work with MARC records at the consortial level, so we assign the UPDATE_MARC, CREATE_MARC, and IMPORT_MARC permissions at depth 0:

```
WITH pgt AS (
  SELECT id
  FROM permission.grp_tree
  WHERE name = 'Student Catalogers'
)
INSERT INTO permission.grp_perm_map (grp, perm, depth)
SELECT pgt.id, ppl.id, 0
FROM permission.perm_list ppl, pgt
WHERE ppl.code IN ('UPDATE_MARC', 'CREATE_MARC', 'IMPORT_MARC');
```

Similarly, we want to give the Student Circulators group the abillity to check out copies and record in-house uses at the system level, so we assign the COPY_CHECKOUT and CREATE_IN_HOUSE_USE permissions at depth 1 (overriding the same *Staff* permissions that were granted only at depth 2):

```
WITH pgt AS (
  SELECT id
  FROM permission.grp_tree
  WHERE name = 'Student Circulators'
) INSERT INTO permission.grp_perm_map (grp, perm, depth)
SELECT pgt.id, ppl.id, 1
FROM permission.perm_list ppl, pgt
WHERE ppl.code IN ('COPY_CHECKOUT', 'CREATE_IN_HOUSE_USE');
```

Finally, we want to add our students to the groups. The request may arrive in your inbox from the library along the lines of "Please add Mint Julep as a Student Cataloger, Bloody Caesar as a Student Circulator, and Grass Hopper as a Student Cataloguer / Circulator; I've already created their accounts and given them a work organizational unit." You can translate that into the following SQL to add the users to the pertinent permission groups, adjusting for the inevitable typos in the names of the users.

First, add our Student Cataloger:

```
WITH pgt AS (
  SELECT id FROM permission.grp_tree
  WHERE name = 'Student Catalogers'
)
INSERT INTO permission.usr_grp_map (usr, grp)
SELECT au.id, pgt.id
FROM actor.usr au, pgt
WHERE first_given_name = 'Mint' AND family_name = 'Julep';
```

Next, add the Student Circulator:

```
 WITH pgt AS (
  SELECT id FROM permission.grp_tree
  WHERE name = 'Student Circulators'
)
INSERT INTO permission.usr_grp_map (usr, grp)
SELECT au.id, pgt.id
FROM actor.usr au, pgt
WHERE first_given_name = 'Bloody' AND family_name = 'Caesar';
```

Finally, add the all-powerful Student Cataloger / Student Circulator:

```
 WITH pgt AS (
  SELECT id FROM permission.grp_tree
  WHERE name IN ('Student Catalogers', 'Student Circulators')
)
INSERT INTO permission.usr_grp_map (usr, grp)
SELECT au.id, pgt.id
FROM actor.usr au, pgt
WHERE first_given_name = 'Grass' AND family_name = 'Hopper';
```

While adopting this role-based approach might seem labour-intensive when applied to a handful of students in this example, over time it can help keep the permission profiles of your system relatively simple in comparison to the alternative approach of rapidly reproducing permission groups, overlapping permissions, and permissions granted on a one-by-one basis to individual users.

5. MIGRATING FROM A LEGACY SYSTEM

When you migrate to Evergreen, you generally want to migrate the bibliographic records and copy information that existed in your previous library system. For anything more than a few thousand records, you should import the data directly into the database rather than use the tools in the staff client. While the data that you can extract from your legacy system varies widely, this section assumes that you or members of your team have the ability to write scripts and are comfortable working with SQL to manipulate data within PostgreSQL. If so, then the following section will guide you towards a method of generating common data formats so that you can then load the data into the database in bulk.

MAKING ELECTRONIC RESOURCES VISIBLE IN THE CATALOG

Electronic resources generally do not have any call number or copy information associated with them, and Evergreen enables you to easily make bibliographic records visible in the public catalog within sections of the organizational unit hierarchy. For example, you can make a set of bibliographic records visible only to specific branches that have purchased licenses for the corresponding resources, or you can make records representing publicly available electronic resources visible to the entire consortium.

Therefore, to make a record visible in the public catalog, modify the records using your preferred MARC editing approach to ensure the 856 field contains the following information before loading records for electronic resources into Evergreen:

856 field for electronic resources: indicators and subfields

Attribute	Value	Note
Indicator 1	4	
Indicator 2	0 or 1	
Subfield u	URL for the electronic resource	
Subfield y	Text content of the link	
Subfield z	Public note	Normally displayed after the link
Subfield 9	Organizational unit short name	The record will be visible when a search is performed specifying this organizational unit or one of its children. You can repeat this subfield as many times as you need.

Once your electronic resource bibliographic records have the required indicators and subfields for each 856 field in the record, you can proceed to load the records using either the command-line bulk import method or the **MARC Batch Importer** in the staff client.

MIGRATING YOUR BIBLIOGRAPHIC RECORDS

Convert your MARC21 binary records into the MARCXML format, with one record per line. You can use the following Python script to achieve this goal; just install the *pymarc* library first, and adjust the values of the *input* and *output* variables as needed.

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
import codecs
import pymarc

input = 'records_in.mrc'
output = 'records_out.xml'

reader = pymarc.MARCReader(open(input, 'rb'),
to_unicode=True)
writer = codecs.open(output, 'w', 'utf-8')
for record in reader:
    record.leader = record.leader[:9] + 'a' +
record.leader[10:]
    writer.write(pymarc.record_to_xml(record) + "\n")
```

Once you have a MARCXML file with one record per line, you can load the records into your Evergreen system via a staging table in your database.

1. Connect to the PostgreSQL database using the *psql* command. For example:

   ```
   psql -U <user-name> -h <hostname> -d <database>
   ```

2. Create a staging table in the database. The staging table is a temporary location for the raw data that you will load into the production table or tables. Issue the following SQL statement from the *psql* command line, adjusting the name of the table from *staging_records_import*, if desired:

   ```
   CREATE TABLE staging_records_import (id BIGSERIAL, dest BIGINT,
   marc TEXT);
   ```

3. Create a function that will insert the new records into the production table and update the *dest* column of the staging table. Adjust "staging_records_import" to match the name of the staging table that you plan to create when you issue the following SQL statement:

   ```
   CREATE OR REPLACE FUNCTION staging_importer() RETURNS VOID AS $$
   DECLARE stage RECORD;
   BEGIN
   FOR stage IN SELECT * FROM staging_records_import ORDER BY id
   LOOP
         INSERT INTO biblio.record_entry (marc, last_xact_id) VALUES
   (stage.marc, 'IMPORT');
         UPDATE staging_records_import SET dest =
   currval('biblio.record_entry_id_seq') WHERE id = stage.id;
      END LOOP;
   END;
   $$ LANGUAGE plpgsql;
   ```

4. Load the data from your MARCXML file into the staging table using the COPY statement, adjusting for the name of the staging table and the location of your MARCXML file:

   ```
   COPY staging_records_import (marc) FROM '/tmp/records_out.xml';
   ```

5. Load the data from your staging table into the production table by invoking your staging function:

   ```
   SELECT staging_importer();
   ```

When you leave out the *id* value for a *BIGSERIAL* column, the value in the column automatically increments for each new record that you add to the table.

Once you have loaded the records into your Evergreen system, you can search for some known records using the staff client to confirm that the import was successful.

MIGRATING YOUR CALL NUMBERS, COPIES,

AND PARTS

Holdings, comprised of call numbers, copies, and parts, are the set of objects that enable users to locate and potentially acquire materials from your library system.

Call numbers connect libraries to bibliographic records. Each call number has a *label* associated with a classification scheme such as a the Library of Congress or Dewey Decimal systems, and can optionally have either or both a label prefix and a label suffix. Label prefixes and suffixes do not affect the sort order of the label.

Copies connect call numbers to particular instances of that resource at a particular library. Each copy has a barcode and must exist in a particular copy location. Other optional attributes of copies include circulation modifier, which may affect whether that copy can circulate or for how long it can circulate, and OPAC visibility, which controls whether that particular copy should be visible in the public catalog.

Parts provide more granularity for copies, primarily to enable patrons to place holds on individual parts of a set of items. For example, an encyclopedia might be represented by a single bibliographic record, with a single call number representing the label for that encyclopedia at a given library, with 26 copies representing each letter of the alphabet, with each copy mapped to a different part such as *A, B, C, ... Z*.

To migrate this data into your Evergreen system, you will create another staging table in the database to hold the raw data for your materials from which the actual call numbers, copies, and parts will be generated.

Begin by connecting to the PostgreSQL database using the *psql* command. For example:

```
psql -U <user-name> -h <hostname> -d <database>
```

Create the staging materials table by issuing the following SQL statement:

```
CREATE TABLE staging_materials (
  bibkey BIGINT,  -- biblio.record_entry_id
  callnum TEXT, -- call number label
  callnum_prefix TEXT, -- call number prefix
  callnum_suffix TEXT, -- call number suffix
  callnum_class TEXT, -- classification scheme
  create_date DATE,
  location TEXT, -- shelving location code
  item_type TEXT, -- circulation modifier code
  owning_lib TEXT, -- org unit code
  barcode TEXT, -- copy barcode
  part TEXT
);
```

For the purposes of this example migration of call numbers, copies, and parts, we assume that you are able to create a tab-delimited file containing values that map to the staging table properties, with one copy per line. For example, the following 5 lines demonstrate how the file could look for 5 different copies, with non-applicable attribute values represented by *\N*, and 3 of the copies connected to a single call number and bibliographic record via parts:

```
1 QA 76.76 A3 \N \N LC 2012-12-05 STACKS BOOK BR1 30007001122620 \N
2 GV 161 V8 Ref. Juv. LC 2010-11-11 KIDS DVD BR2 30007005197073 \N
3 AE 5 E363 1984 \N \N LC 1984-01-10 REFERENCE BOOK BR1
30007006853385 A
3 AE 5 E363 1984 \N \N LC 1984-01-10 REFERENCE BOOK BR1
30007006853393 B
3 AE 5 E363 1984 \N \N LC 1984-01-10 REFERENCE BOOK BR1
30007006853344 C
```

Once your holdings are in a tab-delimited format--which, for the purposes of this example, we will name *holdings.tsv*--you can import the holdings file into your staging table. Copy the contents of the holdings file into the staging table using the *COPY* SQL statement:

```
COPY staging_items (bibkey, callnum, callnum_prefix,
  callnum_suffix, callnum_class, create_date, location,
  item_type, owning_lib, barcode, part) FROM 'holdings.tsv';
```

Generate the copy locations you need to represent your holdings:

```
INSERT INTO asset.copy_location (name, owning_lib)
  SELECT DISTINCT location, 1 FROM staging_materials
  WHERE NOT EXISTS (
    SELECT 1 FROM asset.copy_location
    WHERE name = location
  );
```

Generate the circulation modifiers you need to represent your holdings:

```
INSERT INTO config.circ_modifier (code, name, description,
sip2_media_type)
  SELECT DISTINCT circmod, circmod, circmod, '001'
  FROM staging_materials
  WHERE NOT EXISTS (
    SELECT 1 FROM config.circ_modifier
    WHERE circmod = code
  );
```

Generate the call number prefixes and suffixes you need to represent your holdings:

```
INSERT INTO asset.call_number_prefix (owning_lib, label)
  SELECT DISTINCT aou.id, callnum_prefix
  FROM staging_materials sm
    INNER JOIN actor.org_unit aou
      ON aou.shortname = sm.owning_lib
  WHERE NOT EXISTS (
    SELECT 1 FROM asset.call_number_prefix acnp
    WHERE callnum_prefix = acnp.label
      AND aou.id = acnp.owning_lib
  ) AND callnum_prefix IS NOT NULL;

INSERT INTO asset.call_number_suffix (owning_lib, label)
  SELECT DISTINCT aou.id, callnum_suffix
  FROM staging_materials sm
    INNER JOIN actor.org_unit aou
      ON aou.shortname = sm.owning_lib
  WHERE NOT EXISTS (
    SELECT 1 FROM asset.call_number_suffix acns
    WHERE callnum_suffix = acns.label
      AND aou.id = acns.owning_lib
  ) AND callnum_suffix IS NOT NULL;
```

Generate the call numbers for your holdings:

```
INSERT INTO asset.call_number (
  creator, editor, record, owning_lib, label, prefix, suffix,
label_class
)
  SELECT DISTINCT 1, 1, bibkey, aou.id, callnum, acnp.id, acns.id,
  CASE WHEN callnum_class = 'LC' THEN 1
             WHEN callnum_class = 'DEWEY' THEN 2
  END
  FROM staging_materials sm
    INNER JOIN actor.org_unit aou
      ON aou.shortname = owning_lib
    INNER JOIN asset.call_number_prefix acnp
      ON COALESCE(acnp.label, '') = COALESCE(callnum_prefix, '')
    INNER JOIN asset.call_number_suffix acns
      ON COALESCE(acns.label, '') = COALESCE(callnum_suffix, '')
;
```

Generate the copies for your holdings:

```
INSERT INTO asset.copy (
  circ_lib, creator, editor, call_number, location,
 loan_duration, fine_level, barcode
)
  SELECT DISTINCT aou.id, 1, 1, acn.id, acl.id, 2, 2, barcode
  FROM staging_materials sm
    INNER JOIN actor.org_unit aou
      ON aou.shortname = sm.owning_lib
    INNER JOIN asset.copy_location acl
      ON acl.name = sm.location
    INNER JOIN asset.call_number acn
```

```
      ON acn.label = sm.callnum
  WHERE acn.deleted IS FALSE
;
```

Generate the parts for your holdings. First, create the set of parts that are required for each record based on your staging materials table:

```
INSERT INTO biblio.monograph_part (record, label)
  SELECT DISTINCT bibkey, part
  FROM staging_materials sm
  WHERE part IS NOT NULL AND NOT EXISTS (
    SELECT 1 FROM biblio.monograph_part bmp
    WHERE sm.part = bmp.label
      AND sm.bibkey = bmp.record
  );
```

Now map the parts for each record to the specific copies that you added:

```
INSERT INTO asset.copy_part_map (target_copy, part)
  SELECT DISTINCT acp.id, bmp.id
  FROM staging_materials sm
    INNER JOIN asset.copy acp
      ON acp.barcode = sm.barcode
    INNER JOIN biblio.monograph_part bmp
      ON bmp.record = sm.bibkey
  WHERE part IS NOT NULL
    AND part = bmp.label
    AND acp.deleted IS FALSE
    AND NOT EXISTS (
    SELECT 1 FROM asset.copy_part_map
    WHERE target_copy = acp.id
      AND part = bmp.id
  );
```

At this point, you have loaded your bibliographic records, call numbers, call number prefixes and suffixes, copies, and parts, and your records should be visible to searches in the public catalog within the appropriate organization unit scope.

6. IMPORTING MATERIALS IN THE STAFF CLIENT

Evergreen exists to connect users to the materials represented by bibliographic records, call numbers, and copies -- so getting these materials into your Evergreen system is vital to a successful system. There are two primary means of getting materials into Evergreen:

- The Evergreen staff client offers the **MARC Batch Importer**, which is a flexible interface primarily used for small batches of records;
- Alternately, import scripts can load data directly into the database, which is a highly flexible but much more complex method of loading materials suitable for large batches of records such as the initial migration from your legacy library system.

STAFF CLIENT BATCH RECORD IMPORTS

The staff client has a utility for importing batches of bibliographic and copy records available through **Cataloging > MARC Batch Import/Export**. In addition to importing new records, this interface can be used to match incoming records to existing records in the database, add or overlay MARC fields in the existing record, and add copies to those records.

The MARC Batch Import interface may also be colloquially referred to as "Vandelay" in the Evergreen community, referring to this interface's internals in the system.You will also see this name used in several places in the editor. For instance, when you click on the **Record Match Sets**, the title on the screen will be **Vandelay Match Sets**.

When to use the MARC Batch Importer

- When importing in batches of up to 500 to 1,000 records.
- When you need the system to match those incoming records to existing records and overlay or add fields to the existing record.
- When you need to add copies to existing records in the system.

The MARC Batch Importer should only be used for importing copies that already have barcodes and call numbers. Use **Acquisitions > Load MARC Order Records** to add on-order copies that do not yet have barcodes and call numbers.

Record Match Sets

Click the **Record Match Sets** button to identify how Evergreen should match incoming records to existing records in the system.

These record match sets can be used when importing records through the MARC Batch Importer or when importing order records through the Acquisitions Load MARC Order Records interface.

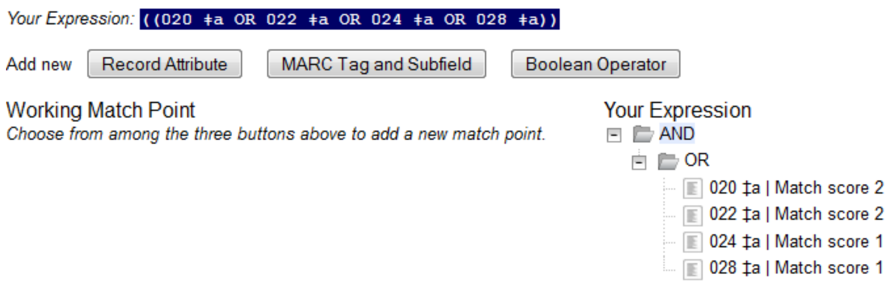
Common match points used when creating a match set include:

- MARC tag 020a (ISBN)
- MARC tag 022a (ISSN)
- MARC tag 024a (UPC)
- MARC tag 028a (Publisher number)

Create Match Sets

1. On the **Record Match Sets** screen, click **New Match Set** to create a set of record match points. Give the set a **Name**. Assign the **Owning Library** from the dropdown list. The **Match Set Type** should remain as **biblio**. Click **Save**.
2. If you don't see your new set in the list, in the upper left corner of the staff client window, click the **Reload** button.
3. If you had to reload, click the **Record Match Sets** button to get back to that screen. Find your new set in the list and click its name. (The name will appear to be a hyperlink.) This will bring you to the **Vandelay Match Set Editor**.
4. Create an expression that will define the match points for the incoming record. You can choose from two areas to create a match: Record Attribute (MARC fixed fields) or MARC Tag and Subfield. You can use the Boolean operators AND and OR to combine these elements to create a match set.
5. When adding a Record Attribute or MARC tag/subfield, you also can enter a **Match Score**. The Match Score indicates the relative importance of that match point as Evergreen evaluates an incoming record against an existing record. You can enter any integer into this field. The number that you enter is only important as it relates to other match points.

   Recommended practice is that you create a match score of one (1) for the least important match point and assign increasing match points to the power of 2 to working points in increasing importance.
6. After creating a match point, drag the completed match point under the folder with the appropriately-named Boolean folder under the Expression tree.


Your Expression: ((020 ‡a OR 022 ‡a OR 024 ‡a OR 028 ‡a))

Working Match Point

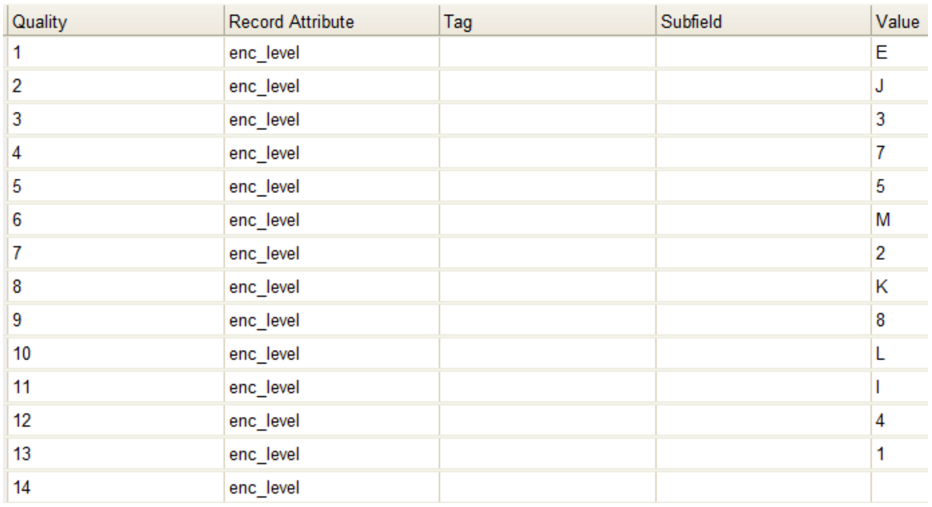
Choose from among the three buttons above to add a new match point.

Your Expression
- AND
  - OR
    - 020 ‡a | Match score 2
    - 022 ‡a | Match score 2
    - 024 ‡a | Match score 1
    - 028 ‡a | Match score 1

7. Click **Save Changes to Expression**.

Quality Metrics

- Quality metrics provide a mechanism for Evergreen to measure the quality of records and to make importing decisions based on quality.
- Metrics are configured in the match set editor.
- Quality metrics are not required when creating a match set.
- You can use a value in a record attribute (MARC fixed fields) or a MARC tag as your quality metric.
- The encoding level record attribute can be one indicator of record quality.

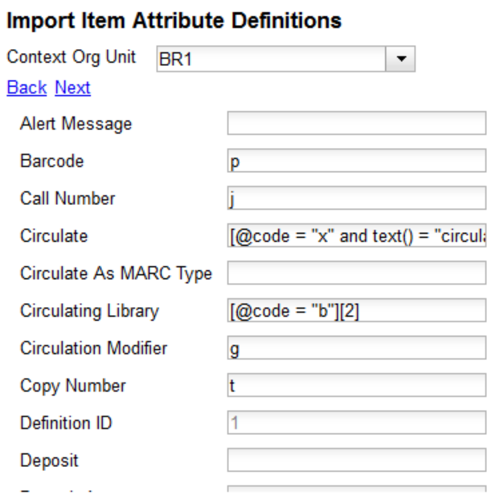


Quality	Record Attribute	Tag	Subfield	Value
1	enc_level			E
2	enc_level			J
3	enc_level			3
4	enc_level			7
5	enc_level			5
6	enc_level			M
7	enc_level			2
8	enc_level			K
9	enc_level			8
10	enc_level			L
11	enc_level			I
12	enc_level			4
13	enc_level			1
14	enc_level			

Import Item Attributes

If you are importing copies with your records, you will need to map the data in your holdings tag to fields in the copy record. Click the **Import Item Attributes** button to map this information.

1. Click the **New Definition** button to create a new mapping for the holdings tag.
2. Use the **Tag** field to identify the MARC tag that contains your holdings information.
3. Add the subfields that contain specific copy information to the appropriate copy field.
4. Select the **Keep** box if Evergreen should keep this holdings tag in the record after it is imported. Otherwise, Evergreen will remove this holdings tag.
5. At a minimum, you should add a name for the definition, identify the MARC holdings tag, and add the subfields that identify the circulating library, the owning library, the call number and the barcode.


Import Item Attribute Definitions

Context Org Unit: BR1

Back Next

Alert Message	
Barcode	p
Call Number	j
Circulate	[@code = "x" and text() = "circul
Circulate As MARC Type	
Circulating Library	[@code = "b"][2]
Circulation Modifier	g
Copy Number	t
Definition ID	1
Deposit	

Overlay/Merge Profiles

If Evergreen finds a match for an incoming record in the database, you need to identify which fields should be replaced, which should be preserved, and which should be added to the record. Click the **Merge/Overlay Profiles** button to create a profile that contains this information.

These overlay/merge profiles can be used when importing records through the MARC Batch Importer or when importing order records through the Acquisitions Load MARC Order Records interface.

Evergreen comes pre-installed with two default profiles:

- **Default merge** - No fields from incoming record are added to match. This profile is useful for item loads or for order record uploads.
- **Default overlay** - Incoming record will replace existing record.

You can customize the overlay/merge behavior with a new profile by clicking the **New Merge Profile** button. Available options for handling the fields include:

- **Preserve specification** - fields in the existing record that should be preserved.

- **Replace specification** - fields in existing record that should be replaced by those in the incoming record.
- **Add specification** - fields from incoming record that should be added to existing record (in addition to any already there.)
- **Remove specification** - fields that should be removed from incoming record.

You can add multiple tags to these specifications, separating each tag with a comma.

Importing the records

After making the above configurations, you are now ready to import your records.

1. Click the **Import Records** button
2. Provide a unique name for the queue where the records will be loaded
3. Identify the match set that should be used for matching
4. If you are importing copies, identify the Import Item Attributes definition in the **Holdings Import Profile**
5. Select a record source
6. Select the overlay/merge profile that defines which fields should be replaced, added or preserved
7. Identify which records should be imported, the options are:

- **Import Non-Matching Records** will automatically import records that have no match in the system
- **Merge on Exact Match** will automatically import records that match on the 901c (record ID)
- **Merge on Single Match** will automatically import records when there is only one match in the system
- **Merge on Best Match** will automatically import records for the best match in the system; the best match will be determined by the combined total of the records match point scores
  You do not need to select any of these import options at this step. You may also opt to review the records first in the import queue and then import them.

- **Best Single Match Minimum Quality Ratio** should only be changed if quality metrics were used in the match set

- Set to 0.0 to import a record regardless of record quality
- Set to 1.0 if the incoming record must be of equal or higher quality than the existing record to be imported
- Set to 1.1 if the incoming record must be of higher quality than the existing record to be imported

- **Insufficient Quality Fall-Through Profile**can also be used with quality metrics. If an incoming record does not meet the standards of the minimum quality ratio, you can identify a back-up merge profile to be used for those records. For example, you may want to use the default overlay profile for high-quality records but use the default merge profile for lower quality records.

7. ORDERING MATERIALS

Acquisitions allows you to order materials, track the expenditure of your collections funds, track invoices and set up policies for manual claiming. In this chapter, we're going to be describing how to use the most essential functions of acquisitions in the Evergreen system.

WHEN SHOULD LIBRARIES USE ACQUISITIONS?

- When you want to track spending of your collections budget.
- When you want to use Evergreen to place orders electronically with your vendors.
- When you want to import large batches of records to quickly get your on-order titles into the system.

If your library simply wants to add on-order copies to the catalog so that patrons can view and place holds on titles that have not yet arrived, acquisitions may be more than you need. Adding those on-order records via cataloging is a simpler option that works well for this use case.

Below are the basic administrative settings to be configured to get started with acquisitions. At a minimum, a library must configure **Funding Sources**, **Funds**, and **Providers** to use acquisitions.

MANAGING FUNDS

Funding Sources (Required)

Funding sources allow you to specify the sources that contribute monies to your fund(s). You can create as few or as many funding sources as you need. These can be used to track exact amounts for accounts in your general ledger.

Example funding sources might be:

- A municipal allocation for your materials budget;
- A trust fund used for collections;
- A revolving account that is used to replace lost materials;
- Grant funds to be used for collections.

Funding sources are not tied to fiscal or calendar years, so you can continue to add money to the same funding source over multiple years, e.g. County Funding. Alternatively, you can name funding sources by year, e.g. County Funding 2010 and County Funding 2011, and apply credits each year to the matching source.

1. To create a funding source, select **Admin** > **Server Administration** > **Acquisitions** > **Funding Source**. Click the **New Funding Source** button. Give the funding source a name, an owning library, and code. You should also identify the type of currency that is used for the fund.
2. You must add money to the funding source before you can use it. Click the hyperlinked name of the funding source and then click the **Apply Credit** button. Add the amount of funds you need to add. The **Note** field is optional.

Funds (Required)

Funds allow you to allocate credits toward specific purchases.

They typically are used to track spending and purchases for specific collections. Some libraries may choose to define very broad funds for their collections (e.g. children's materials, adult materials) while others may choose to define more specific funds (e.g. adult non-fiction DVD's for BR1).

If your library does not wish to track fund accounting, you can create one large generic fund and use that fund for all of your purchases.

1. To create a fund, select **Admin** > **Server Administration** > **Acquisitions** > **Funds**. Click the **New Fund** button. Give the fund a name and code.
2. The **Year** can either be the fiscal or calendar year for the fund.
3. If you are a multi-branch library that will be ordering titles for multiple branches, you should select the system as the owning **Org Unit**, even if this fund will only be used for collections at a specific branch. If you are a one-branch library or if your branches do their own ordering, you can select the branch as the owning **Org Unit**.
4. Select the **Currency Type** that will be used for this fund.
5. You must select the **Active** checkbox to use the fund.
6. Enter a **Balance Stop Percent**. The balance stop percent prevents you from making purchases when only a specified amount of the fund remains. For example, if you want to spend 95 percent of your funds, leaving a five percent balance in the fund, then you would enter 95 in the field. When the fund reaches its balance stop percent, it will appear in red when you apply funds to copies.
7. Enter a **Balance Warning Percent**. The balance warning percent gives you a warning that the fund is low. You can specify any percent. For example, if you want to spend 90 percent of your funds and be warned when the fund has only 10 percent of its balance remaining, then enter 90 in the field. When the fund reaches its balance warning percent, it will appear in yellow when you apply funds to copies.
8. Check the **Propagate** box to propagate funds. When you propagate a fund, the system will create a new fund for the following fiscal year with the same parameters as your current fund. All of the settings transfer except for the year and the amount of money in the fund. Propagation occurs during the fiscal year close-out operation.
9. Check the **Rollover** box if you want to roll over remaining encumbrances and funds into the same fund next year. If you need the ability to roll over encumbrances without rolling over funds, go to the **Library Settings Editor** (**Admin** > **Local Administration** > **Library Settings Editor**) and set **Allow funds to be rolled over without bringing the money along** to **True**.
10. You must add money to the fund before you can begin using it. Click the hyperlinked name of the fund. Click the **Create Allocation** button. Select a **Funding Source** from which the allocation will be drawn and then enter an amount for the allocation. The **Note** field is optional.

Fund Tags (Optional)

You can apply tags to funds so that you can group funds for easy reporting. For example, you have three funds for children's materials: Children's Board Books, Children's DVDs, and Children's CDs. Assign a fund tag of *children's* to each fund. When you need to report on the amount that has been spent on all children's materials, you can run a report on the fund tag to find total expenditures on children's materials rather than reporting on each individual fund.

1. To create a fund tag, select **Admin** > **Server Administration** > **Acquisitions** > **Fund Tags**. Click the the **New Fund Tag** button. Select a owning library and add the name for the fund tag.
2. To apply a fund tag to a fund, select **Admin** > **Server Administration** > **Acquisitions** > **Funds**. Click on the hyperlinked name for the fund. Click the **Tags** tab and then click the **Add Tag** button. Select the tag from the dropdown menu.

ORDERING

Providers (Required)

Providers are the vendors from whom you order titles.

1. To add a provider record, select **Admin** > **Server Administration** > **Acquisitions** > **Providers**.
2. Enter information about the provider. At a minimum, you need to add a **Provider Name**, **Code**, **Owner**, and **Currency**. You also need to select the **Active** checkbox to use the provider.

Distribution Formulas (Optional)

If you are ordering for a multi-branch library system, distribution formulas are a useful way to specify the number of copies that should be distributed to specific branches and copy locations.

1. To create a distribution formula, select **Admin** > **Server Administration** > **Acquisitions** > **Distribution Formulas**. Click the **New Formula** button. Enter the formula name and select the owning library. Ignore the **Skip Count** field.
2. Click **New Entry**. Select an Owning Library from the drop down menu. This indicates the branch that will receive the items.
3. Select a Shelving Location from the drop down menu.
4. In the Item Count field, enter the number of items that should be distributed to that branch and copy location. You can enter the number or use the arrows on the right side of the field.
5. Keep adding entries until the distribution formula is complete.

Helpful acquisitions Library Settings

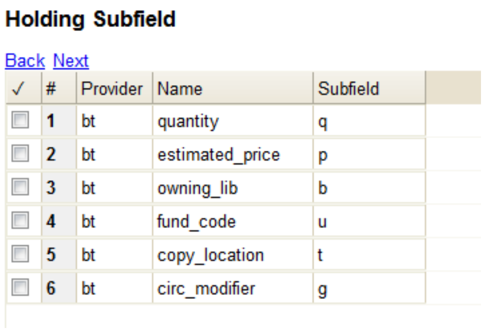
There are several acquisitions Library Settings available that will help with acquisitions workflow. These settings can be found at **Admin** > **Local Administration > Library Settings Editor**.

- Default circulation modifier - Automatically applies a default circulation modifier to all of your acquisitions copies. Useful if you use a specific circulation modifier for on-order copies.
- Default copy location - Automatically applies a default copy location (e.g. On Order) to acquisitions copies.
- Temporary barcode prefix - Applies a unique prefix to the barcode that is automatically generated during the acquisitions process.
- Temporary call number prefix - Applies a unique prefix to the start of the call number that is automatically generated during the acquisitions process.

Preparing for order record loading

If your library is planning to upload order records in a batch, you need to add some information to your provider records so that Evergreen knows how to map the copy data contained in the order record.

1. Retrieve the record for the provider that has supplied the order records by selecting **Admin** > **Server Administration** > **Acquisitions > Providers**. Click on the hyperlinked Provider name.
2. In the top frame, add the MARC tag that contains your holdings data in the **Holdings Tag** field (this tag can also be entered at the time you create the provider record.)
3. To map the tag's subfields to the appropriate copy data, click the **Holding Subfield** tab. Click the **New Holding Subfield** button and select the copy data that you are mapping. Add the subfield that contains that data and click **Save**.



Holding Subfield

Back Next

✓	#	Provider	Name	Subfield
☐	1	bt	quantity	q
☐	2	bt	estimated_price	p
☐	3	bt	owning_lib	b
☐	4	bt	fund_code	u
☐	5	bt	copy_location	t
☐	6	bt	circ_modifier	g

4. If your vendor is sending other data in a MARC tag that needs to be mapped to a field in acquisitions, you can do so by clicking the **Attribute Definitions** tab. As an example, if you need to import the **PO Name**, you could set up an attribute definition by adding an XPath similar to:

```
code => purchase_order
xpath => //*[@tag="962"]/*[@code="p"]
Is Identifier => false
```

   where 962 is the holdings tag and p is the subfield that contains the PO Name.

Preparing to send electronic orders from Evergreen

If your library wants to transmit electronic order information to a vendor, you will need to configure your server to use EDI. You need to install the EDI translator and EDI scripts on your server by following the instructions in the Evergreen 2.3 documentation. (http://docs.evergreen-ils.org/2.3/_installation.html)

Configure your provider's EDI information by selecting **Admin > Server Administration** > **Acquisitions** > **EDI Accounts**. Give the account a name in the **Label** box.

1. **Host** is the vendor-assigned FTP/SFTP/SSH hostname.
2. **Username** is the vendor-assigned FTP/SFTP/SSH username.
3. **Password** is the vendor-assigned FTP/SFTP/SSH password.
4. **Account** is the vendor-assigned account number associated with your organization.
5. **Owner** is the organizational unit who owns the EDI account
6. **Last Activity** is the date of last activity for the account
7. **Provider** is a link to the codes for the Provider record.
8. **Path** is the path on the vendor's server where Evergreen will send its outgoing .epo files.
9. **Incoming Directory** is the path on the vendor's server where incoming .epo files are stored.
10. **Vendor Account Number** is the Vendor assigned account number.
11. **Vendor Assigned Code** is usually a sub-account designation. It can be used with or without the Vendor Account Number.

You now need to add this **EDI Account** and the **SAN** code to the provider's record.

1. Select **Admin** > **Server Administration** > **Acquisitions** > **Providers**.
2. Click the hyperlinked Provider name.
3. Select the account you just created in the **EDI Default** field.
4. Add the vendor-provided SAN code to the **SAN** field.

The last step is to add your library's SAN code to Evergreen.

1. Select **Admin** > **Server Administration** > **Organizational Units**.
2. Select your library from the organizational hierarchy in the left pane.
3. Click the **Addresses** tab and add your library's SAN code to the **SAN** field.

FINDING AND BORROWING MATERIALS

8. DESIGNING YOUR CATALOG

When people want to find things in your Evergreen system, they will check the catalog. In Evergreen, the catalog is made available through a web interface, called the *OPAC* (Online Public Access Catalog). In the latest versions of the Evergreen system, the OPAC is built on a set of programming modules called the Template Toolkit. You will see the OPAC sometimes referred to as the *TPAC*.

In this chapter, we'll show you how to customize the OPAC, change it from its default configuration, and make it your own.

CONFIGURING AND CUSTOMIZING THE PUBLIC INTERFACE

The public interface is referred to as the TPAC or Template Toolkit (*TT*) within the Evergreen community. The template toolkit system allows you to customize the look and feel of your OPAC by editing the template pages (.tt2) files as well as the associated style sheets.

Locating the default template files

The default URL for the TPAC on a default Evergreen system is *http://localhost/eg/opac/home* (adjust *localhost* to match your hostname or IP address).

The default template file is installed in */openils/var/templates/opac*.

You should generally avoid touching the installed default template files, unless you are contributing changes for Evergreen to adopt as a new default. Even then, while you are developing your changes, consider using template overrides rather than touching the installed templates until you are ready to commit the changes to a branch. See below for information on template overrides.

Mapping templates to URLs

The mapping for templates to URLs is straightforward. Following are a few examples, where *<templates>* is a placeholder for one or more directories that will be searched for a match:

- *http://localhost/eg/opac/home ⇒ /openils/var/<templates>/opac/home.tt2*
- *http://localhost/eg/opac/advanced ⇒ /openils/var/<templates>/opac/advanced.tt2*
- *http://localhost/eg/opac/results ⇒ /openils/var/<templates>/opac/results.tt2*

The template files themselves can process, be wrapped by, or include other template files. For example, the *home.tt2* template currently involves a number of other template files to generate a single HTML file.

Example Template Toolkit file: *opac/home.tt2*.

```
[%  PROCESS "opac/parts/header.tt2";
    WRAPPER "opac/parts/base.tt2";
    INCLUDE "opac/parts/topnav.tt2";
    ctx.page_title = l("Home") %]
    <div id="search-wrapper">
      [% INCLUDE "opac/parts/searchbar.tt2" %]
    </div>
    <div id="content-wrapper">
        <div id="main-content-home">
             <div class="common-full-pad"></div>
             [% INCLUDE "opac/parts/homesearch.tt2" %]
             <div class="common-full-pad"></div>
        </div>
     </div>
[% END %]
```

Note that file references are relative to the top of the template directory.

How to override template files

Overrides for template files or TPAC pages go in a directory that parallels the structure of the default templates directory. The overrides then get pulled in via the Apache configuration.

The following example demonstrates how to create a file that overrides the default "Advanced search page" (*advanced.tt2*) by adding a new *templates_custom* directory and editing the new file in that directory.

```
bash$ mkdir -p /openils/var/templates_custom/opac
bash$ cp /openils/var/templates/opac/advanced.tt2 \
         /openils/var/templates_custom/opac/.
bash$ vim /openils/var/templates_custom/opac/advanced.tt2
```

Configuring the custom templates directory in Apache's eg.conf

You now need to teach Apache about the new custom template directory. Edit */etc/apache2/sites-available/eg.conf* and add the following *<Location /eg>* element to each of the *<VirtualHost>* elements in which you want to include the overrides. The default Evergreen configuration includes a VirtualHost directive for port 80 (HTTP) and another one for port 443 (HTTPS); you probably want to edit both, unless you want the HTTP user experience to be different from the HTTPS user experience.

```
<VirtualHost *:80>
    # <snip>

    # - absorb the shared virtual host settings
    Include eg_vhost.conf
    <Location /eg>
        PerlAddVar OILSWebTemplatePath "/openils/var/templates_custom"
    </Location>

    # <snip>
</VirtualHost>
```

Finally, reload the Apache configuration to pick up the changes. You should now be able to see your change at *http://localhost/eg/opac/advanced* where *localhost* is the hostname of your Evergreen server.

Adjusting colors for your public interface

You may adjust the colors of your public interface by editing the *colors.tt2* file. The location of this file is in */openils/var/templates/opac/parts/css/colors.tt2*. When you customize the colors of your public interface, remember to create a custom file in your custom template folder and edit the custom file and not the file located in you default template.

Adjusting fonts in your public interface

Font sizes can be changed by in the *colors.tt2* file located in */openils/var/templates/opac/parts/css/*. Again, create and edit a custom template version and not the file in the default template.

Other aspects of fonts such as the default font family can be adjusted in */openils/var/templates/opac/css/style.css.tt2.*

Media file locations in the public interface

The media files—mostly PNG images—used by the default TPAC templates are stored in the repository in *Open-ILS/web/images/* and installed in */openils/var/web/images/*.

Changing some text in the public interface

Out of the box, TPAC includes a number of placeholder text and links. For example, there is a set of links cleverly named Link 1, Link 2, and so on in the header and footer of every page in TPAC. Here is how to customize that for a *custom templates* skin.

To begin with, find the page(s) that contain the text in question. The simplest way to do that is with the *grep -s* command. In the following example, search for files that contain the text "Link 1":

```
bash$ grep -r "Link 1" /openils/var/templates/opac
/openils/var/templates/opac/parts/topnav_links.tt2
4:            <a href="http://example.com">[% l('Link 1') %]</a>
```

Next, copy the file into our overrides directory and edit it with vim.

Copying the links file into the overrides directory.

```
bash$ cp /openils/var/templates/opac/parts/topnav_links.tt2 \
/openils/var/templates_custom/opac/parts/topnav_links.tt2
bash$ vim /openils/var/templates_custom/opac/parts/topnav_links.tt2
```

Finally, edit the link text in *opac/parts/header.tt2*. Content of the *opac/parts/header.tt2* file.

```
<div id="gold-links-holder">
    <div id="gold-links">
        <div id="header-links">
            <a href="http://example.com">[% l('Link 1') %]</a>
            <a href="http://example.com">[% l('Link 2') %]</a>
            <a href="http://example.com">[% l('Link 3') %]</a>
            <a href="http://example.com">[% l('Link 4') %]</a>
            <a href="http://example.com">[% l('Link 5') %]</a>
        </div>
    </div>
</div>
```

For the most part, the page looks like regular HTML, but note the *[%_(" ")%]* that surrounds the text of each link. The *[% ... %]* signifies a TT block, which can contain one or more TT processing instructions. l(" ... "); is a function that marks text for localization (translation); a separate process can subsequently extract localized text as GNU gettext-formatted PO (Portable Object) files.

As Evergreen supports multiple languages, any customization to Evergreen's default text must use the localization function. Also, note that the localization function supports placeholders such as [_1], [_2] in the text; these are replaced by the contents of variables passed as extra arguments to the l() function.

Once the link and link text has been edited to your satisfaction, load the page in a Web browser and see the live changes immediately.

Adding and removing MARC fields from the record details display page

It is possible to add and remove the MARC fields and subfields displayed in the record details page. In order to add MARC fields to be displayed on the details page of a record, you will need to map the MARC code to variables in the */openils/var/templates/opac/parts/misc_util.tt2 file.*

For example, to map the template variable *args.pubdates* to the date of publication MARC field 260, subfield c, add these lines to *misc_util.tt2*:

```
args.pubdates = [];
```

```
FOR sub IN xml.findnodes('//*[@tag="260"]/*[@code="c"]');
    args.pubdates.push(sub.textContent);
END;
args.pubdate = (args.pubdates.size) ? args.pubdates.0 : ''
```

You will then need to edit the */openils/var/templates/opac/parts/record/summary.tt2* file in order to get the template variable for the MARC field to display.

For example, to display the date of publication code you created in the *misc_util.tt2* file, add these lines:

```
[% IF attrs.pubdate; %]
    <span itemprop="datePublished">[% attrs.pubdate | html; %]</span>
 [% END; %]
```

You can add any MARC field to your record details page. Moreover, this approach can also be used to display MARC fields in other pages, such as your results page.

SETTING THE DEFAULT PHYSICAL LOCATION FOR YOUR LIBRARY ENVIRONMENT

physical_loc is an Apache environment variable that sets the default physical location, used for setting search scopes and determining the order in which copies should be sorted. This variable is set in */etc/apache2/sites-available/eg.conf.* The following example demonstrates the default physical location being set to library ID 104:

```
SetEnv physical_loc 104
```

SETTING A DEFAULT LANGUAGE AND ADDING OPTIONAL LANGUAGES

OILSWebLocale adds support for a specific language. Add this variable to the Virtual Host section in */etc/apache2/sites-available/eg.conf.*

OILSWebDefaultLocale specifies which locale to display when a user lands on a page in TPAC and has not chosen a different locale from the TPAC locale picker. The following example shows the *fr_ca* locale being added to the locale picker and being set as the default locale:

```
PerlAddVar OILSWebLocale "fr_ca"
PerlAddVar OILSWebLocale "/openils/var/data/locale/fr-CA.po"
PerlAddVar OILSWebDefaultLocale "fr-CA"
```

Below is a table of the currently supported languages packaged with Evergreen:

Language	Code	PO file
Czech	cs_cz	/openils/var/data/locale/cs-CZ.po
English - Canada	en_ca	/openils/var/data/locale/en-CA.po
English - Great Britain	en_gb	/openils/var/data/locale/en-GB.po
*English - United States	en_us	not applicable
French - Canada	fr_ca	/openils/var/data/locale/fr-CA.po
Portuguese - Brazil	pt_br	/openils/var/data/locale/pt_BR.po
Russian	ru_ru	/openils/var/data/locale/ru_RU.po

* American English is built into Evergreen so you do not need to set up this language and there are no PO files.

EDITING THE FORMATS SELECT BOX OPTIONS IN THE SEARCH INTERFACE.

You may wish to remove, rename or organize the options in the formats select box. This can be accomplished from the staff client.

1. From the staff client, navigate to **Admin > Server Administration > Marc Coded Value Maps**
2. Select *Type* from the **Record Attribute Type** select box.
3. Double click on the format type you wish to edit.

To change the label for the type, enter a value in the **Search Label** field.

To move the option to a top list separated by a dashed line from the others, check the **Is Simple Selector** check box.

To hide the type so that it does not appear in the search interface, uncheck the **OPAC Visible** checkbox.

Changes will be immediate.

ADDING AND REMOVING SEARCH FIELDS IN ADVANCED SEARCH

It is possible to add and remove search fields on the advanced search page by editing the *opac/parts/config.tt2* file in your template directory. Look for this section of the file:

```
search.adv_config = [
    {adv_label => l("Item Type"), adv_attr => ["mattype",
"item_type"]},
    {adv_label => l("Item Form"), adv_attr => "item_form"},
    {adv_label => l("Language"),  adv_attr => "item_lang"},
    {adv_label => l("Audience"),  adv_attr => ["audience_group",
"audience"], adv_break => 1},
    {adv_label => l("Video Format"), adv_attr => "vr_format"},
    {adv_label => l("Bib Level"), adv_attr => "bib_level"},
    {adv_label => l("Literary Form"), adv_attr => "lit_form", adv_break
=> 1},
    {adv_label => l("Search Library"), adv_special => "lib_selector"},
    {adv_label => l("Publication Year"), adv_special => "pub_year"},
    {adv_label => l("Sort Results"), adv_special => "sort_selector"},
];
```

For example, if you delete the line:

```
{adv_label => l("Language"),  adv_attr => "item_lang"},
```

the language field will no longer appear on your advanced search page. Changes will appear immediately after you save your changes.

CHANGING THE DISPLAY OF FACETS AND FACET GROUPS

Facets can be reordered on the search results page by editing the *opac/parts/config.tt2* file in your template directory.

Edit the following section of *config.tt2*, changing the order of the facet categories according to your needs:

```
facet.display = [
    {facet_class => 'author',  facet_order => ['personal',
'corporate']},
    {facet_class => 'subject', facet_order => ['topic']},
    {facet_class => 'series',  facet_order => ['seriestitle']},
    {facet_class => 'subject', facet_order => ['name', 'geographic']}
];
```

You may also change the default number of facets appearing under each category by editing the *facet.default_display_count* value in *config.tt2*. The default value is 5.

INCLUDING EXTERNAL CONTENT IN THE PUBLIC INTERFACE

The public interface allows you to include external services and content in your public interface. These can include book cover images, user reviews, table of contents, summaries, author notes, annotations, user suggestions, series information among other services. Some of these services are free while others require a subscription

The following are some of the external content services which you can configure in Evergreen.

OpenLibrary

The default install of Evergreen includes OpenLibrary book covers. The settings for this are controlled by the <added_content> section of */openils/conf/opensrf.xml*. Here are the key elements of this configuration:

```
<module>OpenILS::WWW::AddedContent::OpenLibrary</module>
```

This section calls the OpenLibrary perl module. If you wish to link to a different book cover service other than OpenLibrary, you must refer to the location of the corresponding Perl module. You will also need to change other settings accordingly.

```
<timeout>1</timeout>
```

Max number of seconds to wait for an added content request to return data. Data not returned within the timeout is considered a failure.

```
<retry_timeout>600</retry_timeout>
```

This setting is the amount of time to wait before we try again.

```
<max_errors>15</max_errors>
```

Maximum number of consecutive lookup errors a given process can have before added content lookups are disabled for everyone.

To adjust the site of the cover image on the record details page edit the *config.tt2* file and change the value of the *record.summary.jacket_size*. The default value is "medium" and the available options are "small", "medium" and "large."

ChiliFresh

ChiliFresh is a subscription-based service which allows book covers, reviews and social interaction of patrons to appear in your catalog. To activate ChiliFresh, you will need to open the Apache configuration file */etc/apache2/eg_vhost.conf* and edit several lines:

1. Uncoment (remove the "#" at the beginning of the line) and add your chilifresh account number:

   ```
   #SetEnv OILS_CHILIFRESH_ACCOUNT
   ```

2. Uncomment this line and add your ChiliFresh Profile:

   ```
   #SetEnv OILS_CHILIFRESH_PROFILE
   ```

3. Uncomment the line indicating the location of the Evergreen javaScript for ChiliFresh:

   ```
   #SetEnv OILS_CHILIFRESH_URL http://chilifresh.com/on-site /js/evergreen.js
   ```

4. Uncomment the line indicating the secure URL for the Evergreen javaScript :

   ```
   #SetEnv OILS_CHILIFRESH_HTTPS_URL https://secure.chilifresh.com/on-
   site/js/evergreen.js
   ```

Content Café

Content Café is a subscription-based service that can add jacket images, reviews, summaries, tables of contents and book details to your records.

In order to activate Content Café, edit the */openils/conf/opensrf.xml file and* change the <*module*> element to point to the ContentCafe Perl Module:

<module>OpenILS::WWW::AddedContent::ContentCafe</module>

To adjust settings for Content Café Edit a couple of fields with the <*ContentCafe*> Section of */openils/conf/opensrf.xml.*

Edit the *userid* and password elements to match the user id and password for your Content Café account.

Change the *return_behavior_on_no_jacket_image* to set the behavior of your service when an image is not available for an item. By default this value is set to *T* which will result in a small image with the text "No Image Available" in place of a book cover. If you set this value to 1 a 1X1 blank image will be in place of a book cover.

Google Analytics

Google Analytics is a free service to collect statisitics for your Evergreen site. In order to use Google Analytics you will first need to set up the service from the Google Analytics website at http://www.google.com/analytics/. To activate Google Analytics you will need to edit *config.tt2* in your template. To enable the service set the value of *google_analytics.enabled* to *true* and change the value of *google_analytics.code* to be the code in your Google Analytics account.

NoveList

Novelist is a subscription based service providing reviews and recommendation for books in you catalog. To activate your Novelist service in Evergreen, open the Apache configuration file */etc/apache2/eg_vhost.conf* and edit the line:

```
#SetEnv OILS_NOVELIST_URL
```

You should use the URL provided by NoveList.

RefWorks

RefWorks is a subscription-based online bibliographic management tool. If you have a RefWorks subscription, you can activate RefWorks in Evergreen by editing the *config.tt2* file located in your template directory. You will need to set the *ctx.refworks.enabled* value to *true*. You may also set the RefWorks UR by changing the *ctx.refworks.url* setting on the same file.

SFX OpenURL resolver

An OpenURL resolver allows you to find electronic resources and pull them into your catalog based on the ISBN or ISSN of the item. In order to use the SFX OpenURL resolver, you will need to subscribe to the Ex Libris SFX service. To activate the service in Evergreen edit the *config.tt2* file in your template. Enable the resolver by changing the value of *openurl.enabled* to *true* and change the *openurl.baseurl* setting to point to the URL of your openURL resolver.

Syndetic Solutions

Syndetic Solutions is a subscription service providing book covers and other data for items in your catalog. In order to activate Syndetic, edit the */openils/conf/opensrf.xml* file and change the *<module>* element to point to the Syndetic Perl Module:

```
<module>OpenILS::WWW::AddedContent::Syndetic</module>
```

You will also need to edit the *<userid>* element to be the user id provided to you by Syndetic.

Then, you will need to uncomment and edit the *<base_url>* element so that it points to the Syndetic service:

```
<base_url>http://syndetics.com/index.aspx</base_url>
```

For changes to be activated for your public interface you will need to restart Evergreen and Apache.

TROUBLESHOOTING TPAC ERRORS

If there is a problem such as a TT syntax error, it generally shows up as an ugly server failure page. If you check the Apache error logs, you will probably find some solid clues about the reason for the failure. For example, in the following example, the error message identifies the file in which the problem occurred as well as the relevant line numbers.

Example error message in Apache error logs:

```
bash# grep "template error" /var/log/apache2/error_log
[Tue Dec 06 02:12:09 2011] [warn] [client 127.0.0.1] egweb: template
error:
 file error - parse error - opac/parts/record/summary.tt2 line 112-121:
  unexpected token (!=)\n  [% last_cn = 0;\n        FOR copy_info IN
  ctx.copies;\n            callnum = copy_info.call_number_label;\n
```

9. BORROWING ITEMS: WHO, WHAT, FOR HOW LONG

Circulation policies pull together user, library, and item data to determine how library materials circulate, such as: which patrons, from what libraries can borrow what types of materials, for how long, and with what overdue fines.

Individual elements of the circulation policies are configured using specific interfaces, and should be configured prior to setting up the circulation policies.

DATA ELEMENTS THAT AFFECT YOUR CIRCULATION POLICIES

There are a few data elements which must be considered when setting up your circulation policies.

Copy data

Several fields set via the copy editor are commonly used to affect the circulation of an item.

- **Circulation modifier** - Circulation modifiers are fields used to control circulation policies on specific groups of items. They can be added to copies during the cataloging process. New circulation modifiers can be created in the staff client by navigating to **Admin** > **Server Administration** > **Circulation Modifiers**.
- **Circulate?** flag - The circulate? flag in the copy editor can be set to False to disallow an item from circulating.
- **Reference?** flag - The reference? flag in the copy editor can also be used as a data element in circulation policies.

Copy Locations

- To get to the Copy Locations Editor, navigate to **Admin** > **Local Administration** > **Copy Locations Editor**.
- When setting up copy locations for your library you can set an entire copy location to allow or disallow circulation.
- Copy locations can also be used as a data element in circulation policies.

User Permission Group

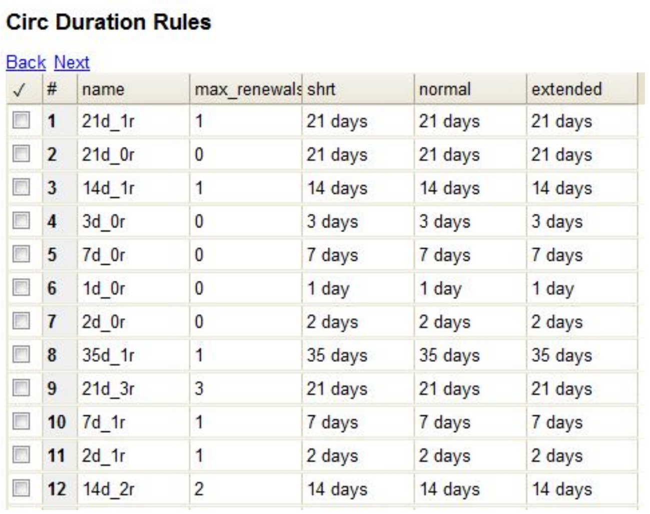
- The user permission group is also commonly used as a data element in circulation policies.

- Other user data that can be used for circulation policies include the juvenile flag in the user record.

CIRCULATION RULES

Loan duration describes the length of time for a checkout. You can also identify the maximum renewals that can be placed on an item.

You can find Circulation Duration Rules by navigating to **Admin** > **Server Administration** > **Circulation Duration Rules**.



Circ Duration Rules

Back Next

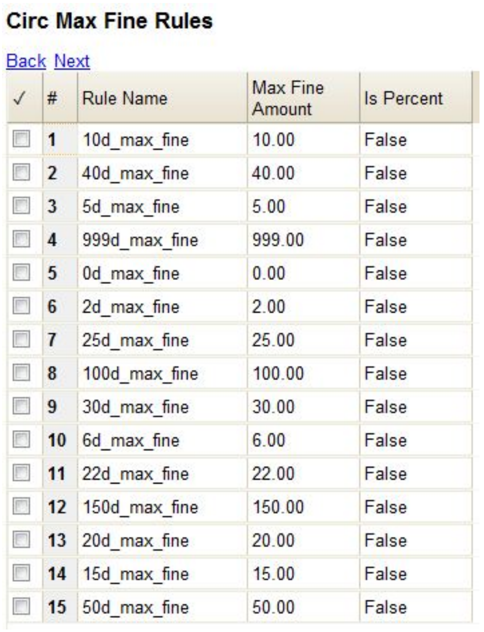
✓	#	name	max_renewals	shrt	normal	extended
☐	1	21d_1r	1	21 days	21 days	21 days
☐	2	21d_0r	0	21 days	21 days	21 days
☐	3	14d_1r	1	14 days	14 days	14 days
☐	4	3d_0r	0	3 days	3 days	3 days
☐	5	7d_0r	0	7 days	7 days	7 days
☐	6	1d_0r	0	1 day	1 day	1 day
☐	7	2d_0r	0	2 days	2 days	2 days
☐	8	35d_1r	1	35 days	35 days	35 days
☐	9	21d_3r	3	21 days	21 days	21 days
☐	10	7d_1r	1	7 days	7 days	7 days
☐	11	2d_1r	1	2 days	2 days	2 days
☐	12	14d_2r	2	14 days	14 days	14 days

Recurring fine describes the amount assessed for daily and hourly fines as well as fines set for other regular intervals. You can also identify any grace periods that should be applied before the fine starts accruing.

You can find Recurring Fine Rules by navigating to **Admin** > **Server Administration** > **Circulation Recurring Fine Rules**.

Max fine describes the maximum amount of fines that will be assessed for a specific circulation. Set the **Use Percent** field to True if the maximum fine should be a percentage of the item's price.

You can find Circ Max Fine Rules by navigating to **Admin** > **Server Administration** > **Circulation Max Fine Rules**.



Circ Max Fine Rules

Back Next

✓	#	Rule Name	Max Fine Amount	Is Percent
☐	1	10d_max_fine	10.00	False
☐	2	40d_max_fine	40.00	False
☐	3	5d_max_fine	5.00	False
☐	4	999d_max_fine	999.00	False
☐	5	0d_max_fine	0.00	False
☐	6	2d_max_fine	2.00	False
☐	7	25d_max_fine	25.00	False
☐	8	100d_max_fine	100.00	False
☐	9	30d_max_fine	30.00	False
☐	10	6d_max_fine	6.00	False
☐	11	22d_max_fine	22.00	False
☐	12	150d_max_fine	150.00	False
☐	13	20d_max_fine	20.00	False
☐	14	15d_max_fine	15.00	False
☐	15	50d_max_fine	50.00	False

These rules generally cause the most variation between organizational units.

Loan duration and recurring fine rate are designed with 3 levels: short, normal, and extended loan duration, and low, normal, and high recurring fine rate. These values are applied to specific items, when copy records are created.

When naming these rules, give them a name that clearly identifies what the rule does. This will make it easier to select the correct rule when creating your circ policies.

Circulation Limit Sets

Circulation Limit Sets allow you to limit the maximum number of copies for different types of materials that a patron can check out at one time. Evergreen supports creating these limits based on circulation modifiers, copy locations, or circ limit groups, which allow you to create limits based on MARC data. The below instructions will allow you to create limits based on circulation modifiers.

- Configure the circulation limit sets by selecting **Admin > Local Administration > Circulation Limit Sets**.
- **Items Out** - the maximum number of items circulated to a patron at the same time.
- **Depth** - Enter the Min Depth, or the minimum depth, in the org tree that Evergreen will consider as valid circulation libraries for counting items out. The min depth is based on org unit type depths. For example, if you want the items in all of the circulating libraries in your consortium to be eligible for restriction by this limit set when it is applied to a circulation policy, then enter a zero (0) in this field.
- **Global** - Check the box adjacent to Global Flag if you want all of the org units in your consortium to be restricted by this limit set when it is applied to a circulation policy. Otherwise, Evergreen will only apply the limit to the direct ancestors and descendants of the owning library.
- **Linked Limit Groups** - add any circulation modifiers, copy locations, or circ limit groups that should be part of this limit set.

Example
Your library (BR1) allows patrons to check out up to 5 videos at one time. This checkout limit should apply when your library's videos are checked out at any library in the consortium. Items with DVD, BLURAY, and VHS circ modifiers should be included in this maximum checkout count.

To create this limit set, you would add 5 to the **Items Out** field, 0 to the **Depth** field and select the **Global Flag**. Add the DVD, BLURAY and VHS circ modifiers to the limit set.

CREATING CIRCULATION POLICIES

Once you have identified your data elements that will drive circulation policies and have created your circulation rules, you are ready to begin creating your circulation policies.

If you are managing a small number of rules, you can create and manage circulation policies in the staff client via **Admin > Local Administration > Circulation Policies**. However, if you are managing a large number of policies, it is easier to create and locate rules directly in the database by updating **config.circ_matrix_matchpoint**.

The **config.circ_matrix_matchpoint** table is central to the configuration of circulation parameters. It collects the main set of data used to determine what rules apply to any given circulation. It is useful for us to think of their columns in terms of “match” columns, those that are used to match the particulars of a given circulation transaction, and “result” columns, those that return the various parameters that are applied to the matching transaction.

- Circulation policies by checkout library or owning library?
    - If your policies should follow the rules of the library that checks out the item, select the checkout library as the **Org Unit (org_unit)**.
    - If your policies should follow the rules of the library that owns the item, select the consortium as the **Org Unit (org_unit)** and select the library as the **Copy Circ Lib (copy_circ_lib)**.
- Renewal policies can be created by setting **Renewals? (is_renewal)** to True.
- You can apply the duration rules, recurring fine rules, maximum fine rules, and circulation sets created in the above sets when creating the circulation policy.

Best practices for creating policies

- Start by replacing the default consortium-level circ policy with one that contains a majority of your libraries' duration, recurring fine, and max fine rules. This first rule will serve as a default for all materials and permission groups.
- If many libraries in your consortium have rules that differ from the default for particular materials or people, set a consortium-wide policy for that circ modifier or that permission group.
- After setting these consortium defaults, if a library has a circulation rule that differs from the default, you can then create a rule for that library. You only need to change the parameters that are different from the default parameters. The rule will inherit the values for the other parameters from that default consortium rule.
- Try to avoid unnecessary repetition.
- Try to get as much agreement as possible among the libraries in your consortium.

Example 1

org_unit	**duration_rule**	**recurring_fine_rule**	**max_fine_rule**
CONS	21_day_2_renew	NONE	NONE
SYS1	NULL	25_cents	3_dollars

In this example, the consortium has decided on a 21_day_2_renew loan rule for general materials, i.e. books, etc. Most members do not charge overdue fines. System 1 charges 25 cents per day to a maximum of $3.00, but otherwise uses the default circulation duration.

Example 2

org_unit	grp	circ_ modifier	circulate	duration_rule	recurring_ fine_rule	max_fine_rule
CONS	Users	book	TRUE	21_day_2_renew	5_cents	2_dollars
CONS	Users	dvd	FALSE	14_day_1_renew	25_cents	5_dollars
CONS	Users	music	TRUE	14_day_2_renew	10_cents	3_dollars

This example includes a basic set of fields and creates a situation where items with a circ modifier of "book" or "music" can be checked out, but "dvd" items will not circulate. The associated rules would apply during checkouts.

Example 3

org_unit	grp	circ_modifier	circulate	duration_ rule	recurring_ fine_rule	max_fine_rule
CONS	Users	book	TRUE	21_day_2_renew	5_cents	2_dollars
CONS	Users	dvd	FALSE	14_day_1_renew	25_cents	5_dollars
SYS1	Adult	dvd	TRUE	14_day_1_renew	25_cents	5_dollars
CONS	Users	music	TRUE	14_day_2_renew	10_cents	3_dollars
SYS2	Users	music	FALSE	14_day_2_renew	10_cents	3_dollars
BR3	Users	music	TRUE	14_day_2_renew	10_cents	3_dollars

This example builds on the earlier example and adds some more complicated options.

It is still true that "book" and "music" items can be checked out, while "dvd" is not circulated. However, now we have added new rules that state that "Adult" patrons of "SYS1" can circulate "dvd" items.

Settings Relevant to Circulation

The following circulation settings, available via **Admin > Local Administration > Library Settings Editor**, can also affect your circulation duration, renewals and fine policy.

- **Auto-Extend Grace Periods** - When enabled, grace periods will auto-extend. By default this will be only when they are a full day or more and end on a closed date, though other options can alter this.
- **Auto-Extending Grace Periods extend for all closed dates** - If enabled and Grace Periods auto-extending is turned on, grace periods will extend past all closed dates they intersect, within hard-coded limits.
- **Auto-Extending Grace Periods include trailing closed dates** - If enabled and Grace Periods auto-extending is turned on, grace periods will include closed dates that directly follow the last day of the grace period.
- **Checkout auto renew age** - When an item has been checked out for at least this amount of time, an attempt to check out the item to the patron that it is already checked out to will simply renew the circulation.
- **Cap Max Fine at Item Price** - This prevents the system from charging more than the item price in overdue fines.
- **Charge fines on overdue circulations when closed** - Normally, fines are not charged when a library is closed. When set to True, fines will be charged during scheduled closings and normal weekly closed days.

10. CONTROLLING HOW HOLDS ARE FULFILLED

Circulation policies control who can borrow materials--but when particular materials are not available at a given library, a patron needs to request those materials by placing a *hold*. Hold policies pull together user, library, and item data to determine how library materials can be requested for and fulfill hold requests, such as: what patrons, from what libraries can request what types of materials, and what copies can fulfill those requests.

Individual elements of the holds policies are configured using specific interfaces, and should be configured prior to setting up the hold policies.

DATA ELEMENTS THAT AFFECT YOUR HOLD POLICIES

There are several data elements that can be modified to adjust your hold policies.

Copy data

When you modify the attributes of a copy in the **Copy Editor**, those attributes can influence hold fulfillment as follows:

- **Circulation Modifier**: Circulation modifiers can control hold policies on specific groups of items. You can create new circulation modifiers in the staff client from **Admin > Server Administration > Circulation Modifiers**.
- **Holdable?** flag: The **Holdable?** flag, if set to *False*, prevents an item from filling a hold.
- **Reference?** flag: The **Reference?** flag, if set to *True*, can affect hold eligibility.
- **Copy age hold protection rule**: This rule generally applies to new materials copy records, and limits where holds can be picked up for a set period of time. You can only apply copy age hold protection rules to individual copies.

Although not a specific element of a hold policy, you can set **Copy Locations** as non-holdable, thereby preventing all items within that location from filling holds.

User data

Permission groups, hold policies, and global flags in combination define the total number of holds that can be placed by members of different user groups.

- **Max holds**: Specifies the total number of holds a member of a particular permission group can place.
- **Max includes frozen**: Enable this option to make frozen (suspended) holds count toward the total number of holds a permission group can place.
- **Requestor Permission Group** and **User Permission Group**: The *requestor* account places the hold request, while the *user* is the account for which the hold has been placed. When a staff user places a hold for a patron, the staff account may have different hold privileges.

Library data

The attributes of libraries, both as actual entities circulating materials and as conceptual organizational units in a hierarchical structure, play a large role in determining which items fill holds, where requested items can be retrieved from, and how quickly requests are filled.

- **User Home Library**: The home library identified in the user's record.
- **Request Library**: The library where the request was placed.
- **Pickup Library**: Where a hold item is to be retrieved.
- **Item Circ Library**: The library that owns the copy.
- **Owning Library**: The library that owns the call number.
- **Transit Range**: The depth within the organization unit hierarchy at which the hold request can travel.
- **Range is from owning library**: Whether or not transit range is from the owner of the copy.

MANAGING HOLD POLICIES

Hold policies are managed in the Staff Client via the **Hold Policy Configuration** interface, accessed from **Admin > Local Admin > Hold Policies**.

Best practices for creating hold policies

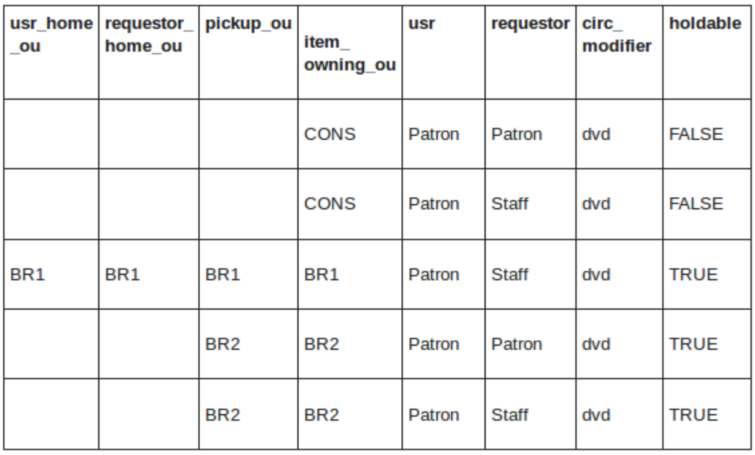
- Identify the parameters that drive your hold policies.
- Attempt to standardize as many hold policies, such as for material types and permission groups, as possible for all organizational units across the consortia.
- Create standard hold policies at the Consortia level to function as default rules.
- Identify where specific organizational units require special rules. Unlike circulation policies, when you add hold policies for child systems or branches, you must identify each parameter for the policy. It will not inherit any values from the default consortium policies.

Simple Example

item_owning_ou	requestor	circ_modifier	holdable
CONS	Users		TRUE
CONS	Users	dvd	FALSE
BR1	Users	music	FALSE
BR1	Adult	music	TRUE
BR2	Adult	dvd	TRUE
BR2	Adult	music	TRUE

At the consortium level, DVDs cannot be placed on hold, but adults can place holds on DVD's owned by BR2.

More Complex Example



usr_home_ou	requestor_home_ou	pickup_ou	item_owning_ou	usr	requestor	circ_modifier	holdable
			CONS	Patron	Patron	dvd	FALSE
			CONS	Patron	Staff	dvd	FALSE
BR1	BR1	BR1	BR1	Patron	Staff	dvd	TRUE
		BR2	BR2	Patron	Patron	dvd	TRUE
		BR2	BR2	Patron	Staff	dvd	TRUE

A set of rules to define holdable behavior for sites BR1 and BR2. Items with a circ modifier of dvd are not allowed to be placed on hold across the CONS level, but then if a staff member places the hold at BR1, the hold is allowed. At BR2, it allows both patrons or staff to place holds on items as long as they choose to pick-up at the owning library.

HOLD TARGETING AND CAPTURE

There are two parallel processes that run to identify potential copies to fulfill hold requests: targeting and opportunistic capture.

- Targeting looks for a copy with "available" or "reshelving" status. Preference is given to copies shelved at the copy's pickup library. Those copies are placed on the library's holds pull list.
- Opportunistic capture checks each copy when it is checked in to determine if it can fill a hold, even if that hold is already on a holds pull list.

By default, opportunistic capture attempts to fill the first hold in the queue where the pickup library matches the checkin library. If there is no hold with that pickup library, it then fills the next hold in a queue.

A library can adjust this behavior by enabling First In First Out (FIFO) holds. In a FIFO environment, the first hold placed is the first hold filled, and so on. Opportunistic capture sends the check-in copy to the next hold in the queue, regardless of the pickup library. You can enable the FIFO policy from **Admin > Local Administration > Library Settings Editor: FIFO**.

Several library settings control the targeting of items for hold capture in **Admin > Local Administration > Library Settings Editor**:

- **Checkout Fills Related Hold**: When a patron checks out an item and they have no holds that directly target the item, the system attempts to find a hold for the patron that could be fulfilled by the checked out item and fulfills it.
- **Target copies for a hold even if copy's circ lib is closed**
- **Target copies for a hold even if copy's circ lib is closed if the circ lib is the hold's pickup lib**
- **Maximum library target attempts**: When this value is set and greater than 0, the system only attempts to find a copy at each possible branch the configured number of times.

AGE HOLD PROTECTION

Age hold protection prevents new items from filling holds requested for pickup at a library other than the owning library for a specified period of time.

You can define the protection period in **Admin > Server Administration > Age Hold Protect Rules**.

The protection period when applied to a copy record can start with the copy record create date (default) or active date. You can change this setting in **Admin > Local Administration > Library Settings Editor: Use Active Date for Age Protection**.

In addition to time period, you can set the *proximity* value to define which organizational units are allowed to act as pickup libraries. The proximity values affect holds as follows:

- "0" allows only holds where pickup library = owning library
- "1" allows holds where pickup library = owning library, parent, and child organizational units
- "2" allows holds where pickup library = owning library, parent, child, and/or sibling organizational units

Age protection only applies to individual copy records. You cannot configure age protection rules in hold policies.

BOUNDARIES

Hold boundaries define which organizational units can fill specific holds. Large consortia containing multiple systems can use hold boundaries to limit transit and delivery costs. There are two types of boundaries, hard and soft.

A hard boundary prevents holds from being filled by copies from organizational units outside of the boundary. If a single potential copy exists and is later changed to a non-holdable status, the hold request will never be filled.

Soft boundaries prioritize targeting for copies from organizational units within the soft boundary first. If no potential copies exist within the soft boundary, the hold targeting algorithm moves up the organization unit hierarchy to the next level until it targets a matching copy or meets a hard boundary.

You can set hold boundaries in **Admin > Local Administration > Library Settings Editor: Hard Boundary** and the corresponding **Soft Boundary** category.

You can also apply hold boundaries to Hold Policies using the **Range is from owning library** toggle and **Transit Range** selection list.

11. MANAGING PATRON PENALTIES AND ALERTS

Normally, a penalty represents a kind of punishment. In Evergreen, however, a *penalty* is an action or alert related to a problem with a patron account. Some classes of Evergreen penalties are related to financial liabilities (excessive fines, excessive money owed for other reasons like lost or damaged materials), while other Evergreen penalties are simply alerts to librarians about problems with a patron account such as an incorrect daytime telephone number or invalid email address.

WHAT ARE PENALTIES?

Library staff with the appropriate permissions can assign penalties to a patron account relating to patron misbehavior in the system. Penalties can range from a simple alert when a patron tries to check out materials to completely blocking a patron's ability to check out materials until that patron pays the fines that she owed to the library.

Libraries commonly want to block a patron's access to borrowing or requesting materials when she hits a certain fine or money owed limit.The Evergreen system refers to these types of penalties as s*tanding penalties*.

CREATING A STANDING PENALTY

We can define standing penalties and what actions the system will take if a penalty reaches a certain threshold.

1. In the upper right corner of the staff client, click **Admin > Local Administration > Standing Penalties**. The Standing Penalty Types window appears.
2. Click **New Penalty Type**. A dialog box appears.
3. In the **name** field, enter *PATRON_EXCEEDS_FINES*.
4. In the **label** field, enter *Patron exceeds fine threshold*.
5. In the **block_list** field, enter *CIRC|FULFILL|HOLD|CAPTURE|RENEW*.
6. (Optional) The **org_depth** field determines at which depth of the organizational unit hierarchy the penalty takes effect.
7. Select the **staff_alert** check box to display an alert when the patron record is viewed in the staff client.
8. Click **Save**.

The **block_list** field defines the list of functions that you want to stop the patron from being able to access. CIRC means circulation. FULFILL means fulfilling hold requests. HOLD means placing holds. CAPTURE means finding available materials in the system for hold requests. RENEW means renewing materials that are checked out. You can choose to enter all of these or only a couple. Items must be separated by the pipe character |. For example, CIRC|RENEW would block the patron from borrowing or renewing but not from placing holds.

CREATING A THRESHOLD FOR THE PENALTY

Now that you have defined a standing penalty for PATRON_EXCEEDS_FINES, you have to tell the system the threshold at which this penalty should be applied. If you do not define a *penalty threshold*, the penalty will never be applied.

1. In the staff client, the upper right corner, click **Admin > Local Administration > Group Penalty Thresholds**. The **Penalty Threshold** window appears.
2. Click **New Penalty Threshold**. A dialog box appears.
3. From the **Group** selection list, choose the group to whi ch you want to apply the threshold. For this example, choose *Patrons*.
4. From the **Org Unit** selection list, choose the level at which the penalty threshold will kick in. Many consortia set a maximum fine limit for all member libraries. For this example, choose the consortial organizational unit. This causes the threshold for your library's patrons to be the same wherever they go in the consortium.
5. From the **Penalty** selection list, choose the penalty that you just created: *PATRON_EXCEEDS_FINES*.
6. In the **Threshold** field, enter *10.00*.
7. Click **Save**.

A threshold of 10.00 will cause the penalty to kick in when the patron's money owed balance reaches or exceeds ten dollars. Your library or your consortium may have a different standard amount. Many penalties in Evergreen do not have penalty thresholds associated with them, but all penalties pertaining to financial matters have penalty thresholds.

DELETING STANDING PENALTIES AND GROUP PENALTY THRESHOLDS

Both the **Standing Penalty** window and the **Penalty Threshold** window contain a check box on the left hand column of the list of penalties or penalty thresholds. If you select that box, then click **Delete Selected**, the associated penalty or penalty threshold is deleted without asking for confirmation. If you accidentally delete a penalty or penalty threshold, you have to recreate it.

CREATING A PENALTY THAT ONLY GENERATES AN ALERT TO THE LIBRARIAN

Evergreen penalties can be used to only generate an alert to the librarian. For example, perhaps your library wants to ensure that every patron has a valid daytime telephone number in their account. Some patron accounts might have been created without a complete patron registration form. This does not represent an emergency but, for the convenience of the library, when a patron with an invalid or incomplete daytime telephone number checks something out, you would like to notify the librarian that the patron's account needs some attention.

1. In the staff client, the upper right corner, click **Admin > Local Administration > Standing Penalties**. The **Standing Penalty Types** window appears.
2. Click **New Penalty Type**. A dialog box appears.
3. In the **name** field, enter *INVALID_PATRON_DAY_PHONE*.
4. In the **label** field, enter *Patron has an invalid daytime phone number*.
5. Leave the **block_list** field empty. You do not want to prevent the patron from using the library; you simply want to generate an alert for the librarian.
6. In the **org_depth** field, enter 0. This depth displays the alert for the entire consortium.
7. Select the **staff_alert** check box to display an alert when the patron record is viewed in the staff client.
8. Click **Save**.

Now, when a patron with a missing daytime telephone number checks something out, the librarian will automatically see an alert message telling them that the patron has an invalid daytime phone number.

12. SENDING REMINDERS TO YOUR USERS

Evergreen allows you to set up automated notifications and actions to respond to events in your system. An example of an event might be an item becoming overdue. An example of an action might be sending an email notification to users.

The **Action Triggers and Notification** module can be found in the staff client by navigating to **Admin** > **Local Administration** > **Notifications / Action Triggers**.

CREATING NOTIFICATIONS

Action Triggers and Notifications are created from within the staff client.

Before creating a new notification, check the existing notifications to see if your desired notification can be cloned from an existing one.

1. From the top menu, select **Admin > Local Administration > Notifications / Action triggers**
2. Click on the **New** button
3. Select an **Owning Library** from the dropdown box
4. Create a unique **Name** for your new action trigger
5. Select the **Hook** from the dropdown box
6. Check the **Enabled** check box
7. Set the **Processing Delay** in the appropriate format. For example, to set the delay to *7 days* to run 7 days from the trigger event or *00:01:00* to run 1 hour after the Processing Delay Context Field.
8. Set the **Processing Delay Context Field** and **Processing Group Context Field**
9. Select the **Validator**, **Reactor**, **Failure Cleanup** and **Success Cleanup** from their respective dropdown boxes
10. Enter text in the **Template** text box if required. These are for email messages. (You may have to scroll down in the staff client window to see this.)
11. Once you are satisfied with your new event trigger, click the **Save** button located at the bottom of the form

CLONING NOTIFICATIONS

Cloning notifications is a great way to create a new notification quickly based on existing ones. To Clone notifications follow these steps:

1. Check the check box next to the action trigger you wish to clone
2. Click **Clone Selected** on the top left of the page.
3. An editing window with open. Notice that the fields will be populated with content from the cloned action trigger. Edit as necessary and give the action trigger a unique Name.

DELETING NOTIFICATIONS

1. Before you delete a notification remember that you can disable a notification so it is inactive without deleting it.
2. Check the check box next to the action trigger you wish to delete
3. Click **Delete Selected** on the top left of the page.

EDITING EMAIL CONTENT FOR NOTIFICATIONS

There are several notifications already created in Evergreen by default, including several for overdue or pre-due events. Text can be edited for email notifications. To edit the text in an existing notification:

1. Navigate to **Admin > Local Administration > Notifications / Action triggers**
2. Open the event by double clicking on it from the list of events.
3. Edit the text in the **Template** field and click **Save**

PROCESSING NOTIFICATIONS

Action triggers and notifications will not be processed until the action trigger runner script is run from the command line or a *cron* job.

To process the action triggers, an Evergreen administrator will need to run the trigger processing script */openils/bin/action_trigger_runner.pl* from the command line. This can and should be be set up as a *cron* job scheduled to run automatically.

This script has a number of important options:

--***osrf-config=<config_file>***: OpenSRF core config file. Defaults to: /openils/conf/opensrf_core.xml

--***custom-filters=<filter_file>***: File containing a JSON Object which describes any hooks that should use a user-defined filter to find their target objects. Defaults to:
/openils/conf/action_trigger_filters.json

--***run-pending***: Run pending events

--***process-hooks:*** Create hook events

--***max-sleep=<seconds>*** When in process-hooks mode, wait up to <seconds> for the lock file to go away. Defaults to 3600 (1 hour).

--***hooks=hook1[,hook2,hook3,...]***: Define which hooks to create events for. If none are defined, it defaults to the list of hooks defined in the --**custom-filters** option.

--***granularity=<label>***: Run events with {label} granularity setting, or no granularity setting

--***granularity-only***: Used in combination with --**granularity**, prevents the running of events with no granularity setting

--***debug-stdout***: Print server responses to stdout (as JSON) for debugging

--***lock-file=<file_name>***: Lock file

For example, to run all pending actions, enter the following command at the command line:

```
perl action_trigger_runner.pl --osrf-
config=/openils/conf/opensrf_core.xml --run-pending --process-hooks
```

If you want to run all actions under a specific hook, enter the following command:

```
 perl action_trigger_runner.pl --osrf-config \
/openils/conf/opensrf_core.xml --hooks checkout.due
```

One of the more useful options is granularity since this goes hand and hand with your *cron* job scheduling. *Granularity* options include *Hourly*, *Daily*, *Weekly*, *Monthly* and *Yearly*.

For the action triggers to be processed, ensure that they are set to **enabled** from the staff client.

CARE AND FEEDING OF EVERGREEN

13. PREPARING FOR DISASTER: SAVING YOUR DATA

Although it might seem pessimistic, spending some of your limited time preparing for disaster is one of the best investments you can make for the long-term health of your Evergreen system. If one of your servers crashes and burns, you want to be confident that you can get a working system back in place -- whether it is your database server that suffers, or an Evergreen application server.

BACKING UP YOUR DATA

At a minimum, you need to be able to recover your system's data from your PostgreSQL database server: patron information, circulation transactions, bibliographic records, and the like. If all else fails, you can at least restore that data to a stock Evergreen system to enable your staff and patrons to find and circulate materials while you work on restoring your local customizations such as branding, colors, or additional functionality. This section describes how to back up your data so that you or a consultant can help you recover from various disaster scenarios.

Creating logical database backups

The simplest method to back up your PostgreSQL data is to use the *pg_dump* utility to create a logical backup of your database. Logical backups have the advantage of taking up minimal space, as the indexes derived from the data are not part of the backup. For example, an Evergreen database with 2.25 million records and 3 years of transactions that takes over 120 GB on disk creates just a 7.0 GB (compressed) backup file. The drawback to this method is that you can only recover the data at the exact point in time at which the backup began; any updates, additions, or deletions of your data since the backup began will not be captured. In addition, when you restore a logical backup, the database server has to recreate all of the indexes--so it can take several hours to restore a logical backup of that 2.25 million record Evergreen database.

As the effort and server space required for logical database backups are minimal, your first step towards preparing for disaster should be to automate regular logical database backups. You should also ensure that the backups are stored in a different physical location, so that if a flood or other disaster strikes your primary server room, you will not lose your logical backup at the same time.

To create a logical dump of your PostgreSQL database:

1. Issue the command to back up your database: *pg_dump -Fc <database-name>* > *<backup-filename>*. If you are not running the command as the *postgres* user on the database server itself, you may need to include options such as *-U <user-name>* and *-h <hostname>* to connect to the database server. You can use a newer version of the PostgreSQL to run *pg_dump* against an older version of PostgreSQL if your client and server operating systems differ. The *-Fc* option specifies the "custom" format: a compressed format that gives you a great deal of flexibility at restore time (for example, restoring only one table from the database instead of the entire schema).
2. If you created the logical backup on the database server itself, copy it to a server located in a different physical location.

You should establish a routine of nightly logical backups of your database, with older logical backups being automatically deleted after a given interval.

Restoring from logical database backups

To increase your confidence in the safety of your data, you should regularly test your ability to restore from a logical backup. Restoring a logical backup that you created using the custom format requires the use of the *pg_restore* tool as follows:

1. On the server on which you plan to restore the logical backup, ensure that you have installed PostgreSQL and the corresponding server package prerequisites. The *Makefile.install* prerequisite installer than came with your version of Evergreen contains an installation target that should satisfy these requirements. Refer to the installation documentation for more details.
2. As the *postgres* user, create a new database using the *createdb* command into which you will restore the data. Base the new database on the *template0* template database to enable the combination of UTF8 encoding and C locale options, and specify the character type and collation type as "C" using the *--lc-ctype* and *--lc-collate* parameters. For example, to create a new database called "testrestore":
   createdb --template=template0 --lc-ctype=C --lc-collate=C testrestore
3. As the *postgres* user, restore the logical backup into your newly created database using the *pg_restore* command. You can use the *-j* parameter to use more CPU cores at a time to make your recovery operation faster. If your target database is hosted on a different server, you can use the *-U <user-name>* and *-h <hostname>* options to connect to that server. For example, to restore the logical backup from a file named *evergreen_20121212.dump* into the "testrestore" database on a system with 2 CPU cores:
   pg_restore -j 2 -d testrestore evergreen_20121212.dump

Creating physical database backups with support for point-in-time recovery

While logical database backups require very little space, they also have the disadvantage of taking a great deal of time to restore for anything other than the smallest of Evergreen systems. Physical database backups are little more than a copy of the database file system, meaning that the space required for each physical backup will match the space used by your production database. However, physical backups offer the great advantage of almost instantaneous recovery, because the indexes already exist and simply need to be validated when you begin database recovery. Your backup server should match the configuration of your master server as closely as possible including the version of the operating system and PostgreSQL.

Like logical backups, physical backups also represent a snapshot of the data at the point in time at which you began the backup. However, if you combine physical backups with write-ahead-log (*WAL*) segment archiving, you can restore a version of your database that represents any point in time between the time the backup began and the time at which the last WAL segment was archived--a feature referred to as point-in-time recovery (PITR). PITR enables you to undo the damage that an accidentally or deliberately harmful UPDATE or DELETE statement could inflict on your production data, so while the recovery process can be complex, it provides fine-grained insurance for the integrity of your data when you run upgrade scripts against your database, deploy new custom functionality, or make global changes to your data.

To set up WAL archiving for your production Evergreen database, you need to modify your PostgreSQL configuration (typically located on Debian and Ubuntu servers in */etc/postgresql/<version>/postgresql.conf*):

1. Change the value of *archive_mode* to *on*
2. Set the value of *archive_command* to a command that accepts the parameters *%f* (representing the file name of the WAL segment) and *%p* (representing the complete path name for the WAL segment, including the file name). You should copy the WAL segments to a remote file system that can be read by the same server on which you plan to create your physical backups. For example, if */data/wal* represents a remote file system to which your database server can write, a possible value of *archive_command* could be: '*test ! -f /data/wal/%f && cp %p /data/wal/%f*'--which effectively tests to see if the destination file already exists, and if it does not, copies the WAL segment to that location. This command can be and often is much more complex (for example, using *scp* or *rsync* to transfer the file to the remote destination rather than relying on a network share), but you can start with something simple.

Once you have modified your PostgreSQL configuration, you need to restart the PostgreSQL server before the configuration changes will take hold:

1. Stop your OpenSRF services.
2. Restart your PostgreSQL server.
3. Start your OpenSRF services and restart your Apache HTTPD server.

To create a physical backup of your production Evergreen database::

1. From your backup server, issue the *pg_basebackup -x -D <data-destination-directory> -U <user-name> -h <hostname> <database-name>* command to create a physical backup of database *<database-name>* on your backup server.

You should establish a process for creating regular physical backups at periodic intervals, bearing in mind that the longer the interval between physical backups, the more WAL segments the backup database will have to replay at recovery time to get back to the most recent changes to the database. For example, to be able to relatively quickly restore the state of your database to any point in time over the past four weeks, you might take physical backups at weekly intervals, keeping the last four physical backups and all of the corresponding WAL segments.

Creating a replicated database

If you have a separate server that you can use to run a replica of your database, consider replicating your database to that server. In the event that your primary database server suffers a hardware failure, having a database replica gives you the ability to fail over to your database replica with very little downtime and little or no data loss. You can also improve the performance of your overall system by directing some read-only operations, such as reporting, to the database replica. In this section, we describe how to replicate your database using PostgreSQL's streaming replication support.

You need to prepare your master PostgreSQL database server to support streaming replicas with several configuration changes. The PostgreSQL configuration file is typically located on Debian and Ubuntu servers at */etc/postgresql/<version>/postgresql.conf*. The PostgreSQL host-based authentication (*pg_hba.conf*) configuration file is typically located on Debian and Ubuntu servers at */etc/postgresql/<version>/pg_hba.conf*. Perform the following steps on your master database server:

1. Turn on streaming replication support. In *postgresql.conf* on your master database server, change *max_wal_senders* from the default value of 0 to the number of streaming replicas that you need to support. Note that these connections count as physical connections for the sake of the *max_connections* parameter, so you might need to increase that value at the same time.
2. Enable your streaming replica to endure brief network outages without having to rely on the archived WAL segments to catch up to the master. In *postgresql.conf* on your master database server, change *wal_keep_segments* to a value such as 32 or 64.
3. Increase the maximum number of log file segments between automatic WAL checkpoints. In *postgresql.conf* on your master database server, change *checkpoint_segments* from its default of 3 to a value such as 16 or 32. This improves the performance of your database at the cost of additional disk space.
4. Create a database user for the specific purpose of replication. As the *postgres* user on the master database server, issue the following commands, where *replicant* represents the name of the new user:
   createuser replicant
   psql -d <database> ALTER ROLE replicant WITH REPLICATION;
5. Enable your replica database to connect to your master database server as a streaming replica. In *pg_hba.conf* on your master database server, add a line to enable the database user *replicant* to connect to the master database server from IP address 192.168.0.164:
   host   replication   replicant   192.168.0.164/32   md5
6. To enable the changes to take effect, restart your PostgreSQL database server.

To avoid downtime, you can prepare your master database server for streaming replication at any maintenance interval; then weeks or months later, when your replica server environment is available, you can begin streaming replication. Once you are ready to set up the streaming replica, perform the following steps on your replica server:

1. Ensure that the version of PostgreSQL on your replica server matches the version running on your master server. A difference in the minor version (for example, 9.1.3 versus 9.1.5) will not prevent streaming replication from working, but an exact match is recommended.
2. Create a physical backup of the master database server.
3. Add a *recovery.conf* file to your replica database configuration directory. This file contains the information required to begin recovery once you start the replica database:
   ```
   # turn on standby mode, disabling writes to the database
   standby_mode = 'on'
    # assumes WAL segments are available at network share /data/wal
   restore_command = 'cp /data/wal/%f %p'
   # connect to the master database to being streaming replication
   primary_conninfo = 'host=kochab.cs.uoguelph.ca user=replicant
   password=<password>
   ```
4. Start the PostgreSQL database server on your replica server. It should connect to the master. If the physical backup did not take too long and you had a high enough value for *wal_keep_segments* set on your master server, the replica should begin streaming replication. Otherwise, it will replay WAL segments until it catches up enough to begin streaming replication.
5. Ensure that the streaming replication is working. Check the PostgreSQL logs on your replica server and master server for any errors. Connect to the replica database as a regular database user and check for recent changes that have been made to your master server.

Congratulations, you now have a streaming replica database that reflects the latest changes to your Evergreen data! Combined with a routine of regular logical and physical database backups and WAL segment archiving stored on a remote server, you have a significant insurance policy for your system's data in the event that disaster does strike.

14. MAXIMIZING PERFORMANCE

Now that you have ensured that the data for your Evergreen system is safe and recoverable in the event of a disaster, you can begin to improve its performance with confidence. While you can improve performance in many ways, this chapter will give you an overview of tuning your ejabberd XMPP server, optimizing the performance of your PostgreSQL database server, and enhancing the configuration of your OpenSRF application servers.

MAKING YOUR EJABBERD XMPP SERVER FASTER

When you installed Evergreen, you had to change a number of *ejabberd* configuration settings to increase its performance. However, some Linux distributions prevent *ejabberd* from using more than one CPU core at a time by default. On a production system running *ejabberd* on Debian or Ubuntu, you can enable the *ejabberd* XMPP server to use more than one CPU core at a time to process requests.

1. Edit */etc/default/ejabberd* and change the *SMP* setting from `#SMP=disable` to `SMP=auto`
2. Restart *ejabberd*.
3. Restart the OpenSRF services.

IMPROVING THE PERFORMANCE OF YOUR POSTGRESQL DATABASE SERVER

Almost every Evergreen operation requires communicating with a PostgreSQL database. If the PostgreSQL database is slow, then Evergreen is slow. This section describes how to ensure that PostgreSQL is as fast as it can be, given the resources that you have available for your Evergreen system.

Install your PostgreSQL database onto its own separate, dedicated hardware

The performance of your database server depends on having access to large amounts of RAM so that it can cache frequently-accessed data in memory and avoid having to read from the disk, which is a much slower operation. Correspondingly, if other software on the system blocks PostgreSQL from accessing the hard drives for read or write operations, then the performance of Evergreen will suffer. Therefore, one of the best options for maximizing the performance of Evergreen is to dedicate a separate server to the PostgreSQL database.

Write transaction logs to a separate physical drive

If your database server has enough disk drives, put the transaction log on a separate physical drive or volume from your data. This prevents writes to the transaction log from blocking reads and writes of your data.

Add RAM to your database server

If your server can cache the entire contents of your database in RAM, then the performance of your database will be many times faster than if it has to occasionally access data from disk. Your database size may not allow this, however; for example, an Evergreen system with 2.25 million bibliographic records currently uses over 120 GB of data. Alternately, your performance can still benefit if you have enough RAM to keep the largest, most frequently accessed database tables cached in memory. The same Evergreen system holds approximately 20 GB of data in the core index tables.

Add CPU cores to your database server

If you have the luxury of having enough RAM to cache your database in memory, then adding CPU cores to your database server can improve performance as each CPU core can handle one concurrent query. For example, if 10 users submit search requests at the same time on your Evergreen server, a database server with only 1 CPU core will finish the first query before it can handle the next query, while a database server with 16 CPU cores would be able to process all of the queries simultaneously.

Tune your PostgreSQL database

In the interests of conserving RAM and CPU resources, the default configuration for PostgreSQL on Linux distributions tends to be quite conservative. While that configuration can support a freshly installed Evergreen system, production Evergreen systems must adjust that configuration to maximize their performance. On most Linux distributions, the PostgreSQL configuration is kept in a file named *postgresql.conf*. The *pgtune* utility can be used to generate some suggestions for your system, or you can start with the following rules of thumb:

- *shared_buffers*: This setting dedicates memory to caching frequently accessed data and blocks the operating system from accessing that memory. Set this to 1/4 of the available RAM on your server.
- *max_connections*: This setting determines how many concurrent physical connections your database server will support. If you are not using a connection pooling solution such as pgBouncer (http://wiki.postgresql.org/wiki/PgBouncer), this setting needs to be more than the total number of maximum children defined in the *opensrf.xml* configuration file for all services that connect to the database, plus extra connections for manual connections to the database and any scripts you might need to run. If this number is too low, your system may run out of available database connections and return random errors. As a caveat, each physical connection increases the maximum amount of memory consumed by PostgreSQL and can lead to running out of memory on the database server.
- *default_statistics_target*: This setting determines how much of the data in each table PostgreSQL samples to determine how to access that data when it processes a query. If the statistics target is too low, PostgreSQL may choose a bad (slow) plan. Set this to 1000.
- *effective_cache_size*: This setting represents how much memory is available to the operating system to cache frequently accessed data, and affects PostgreSQL's choice of access plans. Set this to approximately 60% of the available RAM on your server.

Running reports against a replica database

If you have the luxury of having a replica database server, you can use it for more than just disaster recovery. PostgreSQL versions 9.0 and later support reads against replica databases. In this scenario, you can point the Evergreen reporter service at your replica database for an easy performance win:

- Reports cannot tie up your production database server with long-running queries, which can be a severe problem if a report template contains a particularly complicated set of relationships
- You can reduce the *max_connections* required in your database configuration, freeing up memory for caching data
- Your production database server can cache the data that is most necessary for searches and circulation transactions rather than the data required by arcane reports.

To run your reports against a replica database:

1. In your *opensrf.xml* file, find the *<reporter>* section. When you installed Evergreen, *eg_db_config.pl* will have set the *<database>* and *<state_store>* connection information to connect to your production database server.
2. Change the *<database>* connection information to connect to the replica database.
3. Keep the *<state_store>* connection information pointing at your production database server. The reporter process needs to be able to write to the database to update the status of each scheduled report, and a replica database built on streaming replication is, by definition, a read-only database.
   Using a replication tool like Slony can result in a writable replica database, but that is outside the scope of this document.
4. Restart the Perl OpenSRF services to load the changed configuration.
5. Run *clark-kent.pl* for the report generator to load the new database connection information.

ADDING ANOTHER EVERGREEN APPLICATION SERVER

If you have the resources to add another server to your Evergreen system, consider using it as an additional Evergreen application server to run selected OpenSRF services. At the same time, you can use the divide and conquer strategy to distribute the ejabberd, Apache, and memcached services between your two Evergreen application servers. In the following scenario, we refer to the first Evergreen application server in your system as *app_server_1* and the second Evergreen application server in your system as *app_server_2*.

app_server_1 configuration

We will use *app_server_1* for the following services:

- Apache
- All OpenSRF applications defined in *opensrf.xml*
- A network share serving up */openils* (typically via NFS)
- A network share serving up a writable location for MARC batch import (typically via NFS)

app_server_2 configuration

We will use *app_server_2* for the following services:

- ejabberd
- memcached
- OpenSRF router
- All OpenSRF applications defined in *opensrf.xml*
- A read-only network share from *app_server_1* mounted at */openils*
- A writable network share from *app_server_1* mounted at */nfs-cluster*

Enabling Apache to access the correct memcached server

When the Apache server is not on the same server as your memcached server, you must update the Apache configuration in *eg_vhost.conf* to set the *OSRFTranslatorCacheServer* value to the IP address (or hostname) and port on which the memcached server is available.

Starting and stopping OpenSRF services across multiple servers

Within a single cluster of Evergreen servers, you can only have one OpenSRF router process running at a time. To stop the OpenSRF Perl and C services cleanly, perform the following steps:

1. *app_server_1*: Stop the OpenSRF C services.
2. *app_server_2*: Stop the OpenSRF C services.
3. *app_server_1*: Stop the OpenSRF Perl services.
4. *app_server_2*: Stop the OpenSRF Perl services.
5. *app_server_2*: Stop the OpenSRF Router.

To start the OpenSRF services, perform the following steps:

1. *app_server_2*: Start the OpenSRF Router.
2. *app_server_2*: Start the OpenSRF Perl services.
3. *app_server_1*: Start the OpenSRF Perl services.
4. *app_server_2*: Start the OpenSRF C services.
5. *app_server_1*: Start the OpenSRF C services.

Providing access to the network shares

The */openils* directory on a single-server Evergreen instance is used to read and write many different files. However, on a multiple-server Evergreen instance, administrators typically turn the */openils* directory into a network share that is writable by only one Evergreen application server and read-only for other application servers. This can complicate your configuration significantly, as a default install of OpenSRF and Evergreen attempts to write log and process ID (PID) files to the */openils/var/log* directory.

Similarly, the *MARC Batch Importer* (*open-ils.vandelay*) service needs to be able to store files temporarily in a common directory, such as */nfs-cluster*.

Defining which services to run on each server

The stock *opensrf.xml* configuration file includes a <*hosts*> section at the very end that lists:

- One or more hostnames as XML elements; for example, <*localhost*>...</*localhost*> in the stock configuration file.
- Each hostname contains one <*activeapps*> element, which in turn contains a set of one or more <*appname*> elements naming the OpenSRF applications that should be started on that given host.

15. STAYING CURRENT AND SECURE

When it comes to running an Evergreen system, there are two special areas of concern:

- How and when you decide to upgrade Evergreen software or apply fixes
- How to take care of the actual server(s) that your Evergreen system uses

The following hints to help you cope with these challenges.

UPGRADING THE EVERGREEN SOFTWARE

The Evergreen community at large have agreed upon an upgrade cycle that produces new major releases twice a year, in Spring and Fall. Major releases can contain new features. The community supports each major release with 12 subsequent monthly minor releases that contain only bug fixes, and continues to provide security fixes if necessary for an additional three months after the end of the regular minor bug fix support, for a total of 15 months of support for each major release.

As a general rule, as the Evergreen community releases each new version of the Evergreen software, they also provide a guideline on how to upgrade from the previous release as part of the official Evergreen documentation at http://docs.evergreen-ils.org. Follow the instructions exactly and in the order that they are given--and if you run into a problem, report it to the community with as much detail about the error message or symptoms of the problem as you can.

Keep the Evergreen release schedule in mind when planning your own testing and upgrade schedules. If you participate in testing new Evergreen releases during the release candidate stages, you will prepare your own library for the upgrade process and help flush out any remaining bugs before the major release of the software. This also gives you time to prepare the members of your library for the upcoming changes by giving them the chance, when possible, to familiarize themselves with new features on your test system. You also have the chance to prepare supporting materials, like handouts and other kinds of documentation, to help your users before, during and after each upgrade cycle.

SECURING THE SERVER(S) ON WHICH YOUR EVERGREEN INSTALLATION RUNS

An Evergreen installation requires interaction between many different components and, depending on the size of your consortium and how many servers you have, it can range from quite complex to extremely. That said, there are a number of standard guidelines that you can follow to secure your server.

- Keep your server up-to-date. Apply security updates as soon as possible when they come out to prevent your system from being exposed to a known vulnerability.
- Pay close attention to account administration on the server. Do not give any user on the server more power than they need.
- Disable services that you do not need.
- Pay attention to your system's log files to see what kind of activity is happening and notice anything unusual.
- A central idea to server security is to make it unreasonably difficult for anyone who tries to compromise your system. Let them choose targets more vulnerable than yours.

This topic is very rich and there are many resources available, both in print and on the web. It is worth your time to learn more.

RESOURCES

16. GETTING MORE HELP IN THE COMMUNITY

The Evergreen community is constantly growing. There is already a strong community of enthusiastic contributors and a wide range of ways to connect with them both to get help and to give help.

FOR A LOT OF HELPFUL INFORMATION ON EVERGREEN

The first, best place to start is **The Evergreen Project web site** at http://www.evergreen-ils.org.

There you will find links to information about:

- Current **documentation** - http://docs.evergreen-ils.org/
- The **Evergreen Wiki** - http://www.evergreen.org/dokuwiki/doku.php
- The Evergreen Community **official blog** - http://www.evergreen-ils.org/blog/

And more!

PARTICIPATE MORE ACTIVELY IN THE COMMUNITY, ASK QUESTIONS AND GET ANSWERS

There are quite a few mailing lists that you can join that deal with various aspects of Evergreen. See all the available lists at - http://evergreen-ils.org/listserv.php. A couple good places to start are:

- The general **Evergreen Discussion Group** list to bring up anything related to Evergreen - http://libmail.georgialibraries.org/mailman/listinfo/open-ils-general
- The **Evergreen Documentation Interest Group** list to learn more about the current state of documentation and contribute to documentation efforts - http://list.georgialibraries.org/mailman/listinfo/open-ils-documentation

You can also connect directly with many tech-savvy people working on Evergreen through the **Freenode Internet Relay Chat** (IRC) Network on channel **#evergreen**. The #evergreen channel is not just for developers or system administrators, but is for any user who has questions about Evergreen or who just wants to say "hi".

You may use an IRC client to connect to this channel or you can connect through the web gateway at http://webchat.freenode.net/?channels=evergreen. Activity in the channel is highest from 9 a.m. to 5 p.m. EST.

STAY UP TO DATE ON CURRENT WORK

Visit the Evergreen section of **Launchpad** to track development, reported bugs, fixes, and enhancements. Launchpad is a collaborative software development platform used by Evergreen developers to manage development work being done on the software.

- Go to https://launchpad.net/evergreen

WHAT'S NEXT FOR YOU?

Friendly interaction seems to be a hallmark of members of the Evergreen community. You are welcome. There is a communication channel for you (or more than one) within the community and no doors are closed.

As you begin your work with Evergreen and interact with other members of the community, you will become known and will be valued for what you do.

You will play your part in the ongoing development of the software. Setting up and running an Evergreen system will provide you with many opportunities to connect with other members of the Evergreen community. This may be through reporting bugs, through writing or fixing software, through creating documentation, to becoming an advocate for Evergreen to the wider library world.

17. GLOSSARY

A

Acquisitions
Processes related to ordering materials and managing expenditures.

Apache
Open-source web server software used to serve both static content and dynamic web pages in a secure and reliable way. More information is available at http://apache.org.

B

Balance stop percent
A setting in acquisitions that prevents you from making purchases when only a specified amount of the fund remains.

C

Call number
An item's call number is a string of letters and or numbers that work like map coordinates to describe where in a library a particular item "lives."

Circulation modifiers
Circulation modifiers pull together Loan Duration, Renewal Limit, Fine Level, Max Fine, and Profile Permission Group to create circulation rules for different types of materials. Circulation Modifiers are also used to determine Hold Policies.

Circulating library
The library which has checked out the item.

Circulation library
The library which is the home of the item.

Circulation limit sets
Refines circulation policies by limiting the number of items that users can check out.

Circulation modifiers
Fields used to control circulation policies on specific groups of items.

Copies
Connect call numbers to particular instances of that resource at a particular library. Each copy has a barcode and must exist in a particular copy location.

D

Distribution formulas
Used to specify the number of copies that should be distributed to specific branches and copy locations.

Due date
The due date is the day on or before which an item must be returned to the library in order to avoid being charged an overdue fine.

E

Electronic data interchange (EDI)
Transmission of data between organizations using electronic means. This is used for Acquisitions.

F

FIFO (First In First Out)
In a FIFO environment, holds are filled in the order that they are placed.

Fund tags
Tags used in acquisitions to allow you to group Funds.

Funding sources
Sources of the monies to fund acquisitions of materials.

Funds
Allocations of money used for purchases.

H

Hold boundaries
Define which organizational units are available to fill specific holds.

Holdings import profile
Identifies the Import Item Attributes definition.

Holding subfield
Used in the acquisitions module to map subfields to the appropriate copy data.

I

Import item attributes
Used to map the data in your holdings tag to fields in the copy record during a MARC import.

Insufficient quality fall-through profile
A back-up merge profile to be used for importing if an incoming record does not meet the standards of the minimum quality ratio.

Item barcode
Item barcodes uniquely identify each specific item entered into the Catalog.

J

Jabber
The communications protocol used for client-server message passing within Evergreen. Now known as XMPP (eXtensible Messaging and Presence Protocol), it was originally named "Jabber."

Juvenile flag
User setting used to specify if a user is a juvenile user for circulation purposes.

L

Loan duration
Loan duration (also sometimes referred to as "loan period") is the length of time a given type of material can circulate.

M

MARC
Acronym for *Machine Readable Cataloging.*The MARC formats are standards for the representation and communication of bibliographic and related information in machine-readable form.

MARC batch export

Mass exporting of MARC records out of a library system.

MARC batch import
Mass importing of MARC records into a library system.

MARCXML
Framework for working with MARC data in a XML environment.

Match score
Indicates the relative importance of that match point as Evergreen evaluates an incoming record against an existing record.

Minimum quality ratio
Used to set the acceptable level of quality for a record to be imported.

O

OPAC
Acronym for *Online Public Access Catalog*. An OPAC is an online interface to the database of a library's holdings, used to find resources in their collections. It is possibly searchable by keyword, title, author, subject or call number.

OpenSRF
Acronym for *Open Scalable Request Framework* (pronounced 'open surf'). OpenSRF is a stateful, decentralized service architecture that allows developers to create applications for Evergreen with a minimum of knowledge of its structure.

Organizational units
Organizational Units are the specific instances of the organization unit types that make up your library's hierarchy.

Organization unit type
The organization types in the hierarchy of a library system.

Overlay/merge profiles
During a MARC import this is used identify which fields should be replaced, which should be preserved, and which should be added to the record.

Owning library
The library which has purchased a particular item and created the volume and copy records.

P

Parent organizational unit
An organizational unit one level above whose policies may be inherited by its child units.

Parts
Provide more granularity for copies, primarily to enable patrons to place holds on individual parts of a set of items.

Patron barcode / library card number
Patrons are uniquely identified by their library card barcode number.

Pickup library
Library designated as the location where requested material is to be picked up.

PostgreSQL
A popular open-source object-relational database management system that underpins Evergreen software.

Propagate funds
Create a new fund for the following fiscal year with the same parameters as your current fund.

Providers
Vendors from whom you order your materials. Set in the Acquisition module.

Purchase Order (PO)
A document issued by a buyer to a vendor, indicating types, quantities, and prices of materials.

Q

Quality metrics
Provide a mechanism for Evergreen to measure the quality of records and to make importing decisions based on quality.

R

Record match sets
When importing records, this identifies how Evergreen should match incoming records to existing records in the system.

Recurring fine
Recurring Fine is the official term for daily or other regularly accruing overdue fines.

Rollover
Used to roll over remaining encumbrances and funds into the same fund the following year.

S

Shelving location
Shelving location is the area within the library where a given item is shelved.

SIP
Acronym for *Standard Interchange Protocol*. SIP is a communications protocol used within Evergreen for transferring data to and from other third party devices, such as RFID and barcode scanners that handle patron and library material information. Version 2.0 (also known as "SIP2") is the current standard. It was originally developed by the 3M Corporation.

SRU
Acronym for *Search & Retrieve URL Service*. SRU is a search protocol used in web search and retrieval. It expresses queries in Contextual Query Language (CQL) and transmits them as a URL, returning XML data as if it were a web page. See Also SRW.

Staff client
The graphical user interface used by library workers to interact with the Evergreen system. Staff use the Staff Client to access administration, acquisitions, circulation, and cataloging functions.

Standing penalties
Serve as alerts and blocks when patron records have met certain criteria, commonly excessive overdue materials or fines; standing penalty blocks will prevent circulation and hold transactions.

T

Template Toolkit (TT)
A template processing system written in Perl.

TPAC
The web based public interface in Evergreen written using functionality from the Template Toolkit.

X

XML
Acronym for *eXtensible Markup Language*, a subset of SGML. XML is a set of rules for encoding information in a way that is both human-readable and machine-readable. It is primarily used to define documents but can also be used to define arbitrary data structures. It was originally defined by the World Wide Web Consortium (W3C).

XMPP
The open-standard communications protocol (based on XML) used for client-server message passing within Evergreen. It supports the concept of a consistent domain of message types that flow between software applications, possibly on different operating systems and architectures. More information is available at http://xmpp.org.

See Also: **Jabber**.

xpath
The XML Path Language, a query language based on a tree representation of an XML document. It is used to programmatically select nodes from an XML document and to do minor computation involving strings, numbers and Boolean values. It allows you to identify parts of the XML document tree, to navigate around the tree, and to uniquely select nodes. The currently version is "XPath 2.0". It was originally defined by the World Wide Web Consortium (W3C).

Z

Z39.50
An international standard client–server protocol for communication between computer systems, primarily library and information related systems.

See Also: **SRU**

18. APPENDIX A: SUGGESTED PROFILE PERMISSIONS

There are well over 500 permissions that can be granted to users in a default installation of Evergreen. Below you will find suggested minimums for four different functional user types within a library: circulation, cataloging, local administration, and patron.

These permissions are not hard and fast. You can add to them. You can even create new permissions that aren't in the default list.

GENERAL PRINCIPLES

Access the **Permission Groups** editor from **Admin > Server Administration > Permission Groups**

Permissions must be added to a group one by one. The good news is that you only have to do this once. After this, you will just be editing the permission group, adding or removing single permissions as necessary.

Be careful with Context Location and depth. For many things, a Consortium depth level will be appropriate. For others, only a depth of System will be appropriate (meaning the local library, even if the local library has no branches). For instance, you wouldn't want to give permission to all the librarians in your consortium to delete each others' holdings from the catalog. But you would want a local library to be able to weed her own collection.

CIRCULATION

The basic, useful permissions to add to the permission profile for librarians who will work the circulation desk are:

ABORT_TRANSIT
BAR_PATRON
CANCEL_HOLDS
CHECKIN_BYPASS_HOLD_FULFILL
CIRC_CLAIMS_RETURNED.override
CIRC_OVERRIDE_DUE_DATE
CIRC_PERMIT_OVERRIDE
COPY_ALERT_MESSAGE.override
COPY_BAD_STATUS.override
COPY_CHECKIN
COPY_CHECKOUT
COPY_HOLDS
COPY_IS_REFERENCE.override
COPY_NEEDED_FOR_HOLD.override
COPY_NOT_AVAILABLE.override
COPY_STATUS_LOST.override
COPY_STATUS_MISSING.override
COPY_TRANSIT_RECEIVE
CREATE_BILL
CREATE_CONTAINER
CREATE_CONTAINER_ITEM
CREATE_COPY
CREATE_COPY_NOTE
CREATE_COPY_STAT_CAT
CREATE_COPY_STAT_CAT_ENTRY
CREATE_COPY_TRANSIT
CREATE_HOLD_NOTIFICATION
CREATE_IN_HOUSE_USE
CREATE_MARC
CREATE_MY_CONTAINER
CREATE_NON_CAT_TYPE
CREATE_PATRON_STAT_CAT
CREATE_PATRON_STAT_CAT_ENTRY
CREATE_PAYMENT
CREATE_TRANSACTION
CREATE_TRANSIT
CREATE_USER
CREATE_VOLUME
CREATE_VOLUME_NOTE
DELETE_CONTAINER
DELETE_COPY
DELETE_HOLDS
DELETE_USER
DELETE_VOLUME
HOLD_ITEM_CHECKED_OUT.override
IMPORT_MARC
ITEM_ON_HOLDS_SHELF.override
MARK_ITEM_DAMAGED
MARK_ITEM_MISSING
MARK_ITEM_MISSING_PIECES
MAX_RENEWALS_REACHED.override
MERGE_USERS
MR_HOLDS
OFFLINE_UPLOAD
OFFLINE_VIEW
OPAC_LOGIN
PATRON_EXCEEDS_CHECKOUT_COUNT.override
PATRON_EXCEEDS_FINES.override
PATRON_EXCEEDS_OVERDUE_COUNT.override
PERSISTENT_LOGIN
REGISTER_WORKSTATION
REMOTE_Z3950_QUERY
RENEW_CIRC
RENEW_HOLD_OVERRIDE
REQUEST_HOLDS

SET_CIRC_CLAIMS_RETURNED
SET_CIRC_LOST
SET_CIRC_MISSING
STAFF_LOGIN
TITLE_HOLDS
UNBAR_PATRON
UPDATE_BATCH_COPY
UPDATE_BILL_NOTE
UPDATE_CONTAINER
UPDATE_COPY
UPDATE_COPY_STAT_CAT
UPDATE_COPY_STAT_CAT_ENTRY
UPDATE_HOLD
UPDATE_MARC
UPDATE_NON_CAT_TYPE
UPDATE_ORG_UNIT
UPDATE_PATRON_ACTIVE_CARD
UPDATE_PATRON_PRIMARY_CARD
UPDATE_PATRON_STAT_CAT
UPDATE_PATRON_STAT_CAT_ENTRY
UPDATE_PAYMENT_NOTE
UPDATE_PICKUP_LIB_FROM_TRANSIT
UPDATE_USER
UPDATE_VOLUME
VIEW_BILLING_TYPE
VIEW_CIRCULATIONS
VIEW_CONTAINER
VIEW_COPY_CHECKOUT_HISTORY
VIEW_COPY_NOTES
VIEW_HOLD
VIEW_HOLD_NOTIFICATION
VIEW_HOLD_PERMIT
VIEW_ORG_SETTINGS
VIEW_PERM_GROUPS
VIEW_PERMISSION
VIEW_PERMIT_CHECKOUT
VIEW_REPORT_OUTPUT
VIEW_TITLE_NOTES
VIEW_TRANSACTION
VIEW_USER
VIEW_USER_FINES_SUMMARY
VIEW_USER_TRANSACTIONS
VIEW_VOLUME_NOTES
VOID_BILLING
VOLUME_HOLDS
actor.org_unit.closed_date.create
actor.org_unit.closed_date.delete
actor.org_unit.closed_date.update
group_application.user
group_application.user.patron

CATALOGING

A cataloger will need all the same permissions as a librarian working at Circulation with a few additions. The extra permissions for a cataloger would be:

CREATE_MFHD_RECORD
CREATE_MONOGRAPH_PART
DELETE_MFHD_RECORD
DELETE_MONOGRAPH_PART
DELETE_RECORD
MERGE_AUTH_RECORDS
UPDATE_MFHD_RECORD
UPDATE_MONOGRAPH_PART
UPDATE_RECORD

LOCAL ADMINISTRATOR

A local administrator permission profile shares some of the characteristics of both a Circulation and Cataloging permission profile. There are additional elements that have to do with reporting.

The local administrator basic permission profile can contain the following permissions.

You can add more permissions as you require but it is generally a good idea not to use a local administrator account for day to day work.

CREATE_TRIGGER_VALIDATOR
CREATE_USER
CREATE_USER_GROUP_LINK
CREATE_VOLUME
CREATE_VOLUME_NOTE
DEBUG_CLIENT
DELETE_CONTAINER
DELETE_COPY_NOTE
DELETE_COPY_STAT_CAT
DELETE_COPY_STAT_CAT_ENTRY
DELETE_COPY_STAT_CAT_ENTRY_MAP
DELETE_PATRON_STAT_CAT
DELETE_PATRON_STAT_CAT_ENTRY
DELETE_PATRON_STAT_CAT_ENTRY_MAP
DELETE_TITLE_NOTE
DELETE_TRIGGER_VALIDATOR
DELETE_USER
DELETE_VOLUME
DELETE_VOLUME_NOTE
HOLD_ITEM_CHECKED_OUT.override
IMPORT_MARC
MR_HOLDS
OFFLINE_EXECUTE
OFFLINE_UPLOAD
OFFLINE_VIEW
OPAC_LOGIN
REGISTER_WORKSTATION
REMOTE_Z3950_QUERY
RENEW_CIRC
RENEW_HOLD_OVERRIDE
REQUEST_HOLDS
RUN_REPORTS
SET_CIRC_CLAIMS_RETURNED
SET_CIRC_LOST
SET_CIRC_MISSING
SHARE_REPORT_FOLDER
STAFF_LOGIN
TITLE_HOLDS
UPDATE_BATCH_COPY
UPDATE_CONTAINER
UPDATE_COPY
UPDATE_COPY_STAT_CAT
UPDATE_COPY_STAT_CAT_ENTRY
UPDATE_NON_CAT_TYPE
UPDATE_ORG_SETTING
UPDATE_ORG_UNIT
UPDATE_PATRON_STAT_CAT
UPDATE_PATRON_STAT_CAT_ENTRY
UPDATE_TRIGGER_HOOK
UPDATE_TRIGGER_VALIDATOR
UPDATE_USER
UPDATE_VOLUME
VIEW_BILLING_TYPE
VIEW_CIRCULATIONS
VIEW_CONTAINER
VIEW_COPY_CHECKOUT_HISTORY
VIEW_COPY_NOTES
VIEW_HOLD
VIEW_HOLD_NOTIFICATION
VIEW_HOLD_PERMIT
VIEW_ORG_SETTINGS
VIEW_PERM_GROUPS
VIEW_PERMISSION
VIEW_PERMIT_CHECKOUT
VIEW_REPORT_OUTPUT

VIEW_TITLE_NOTES
VIEW_TRANSACTION
VIEW_TRIGGER_EVENT_DEF
VIEW_USER
VIEW_USER_FINES_SUMMARY
VIEW_USER_TRANSACTIONS
VIEW_VOLUME_NOTES
VOLUME_HOLDS
actor.org_unit.closed_date.create
actor.org_unit.closed_date.delete
actor.org_unit.closed_date.update

A PATRON PERMISSION PROFILE

Patrons don't require many permissions in the system. They do need to be able to place holds, to log in to the OPAC and perform functions related to their account there. This set of permissions will allow a patron to do practically everything they will ever need to do related to their account and using materials.

If you belong to a consortium which allows patrons to use their library card at any library, then the depth of these permissions should be set to Consortium.

ABORT_REMOTE_TRANSIT
COPY_CHECKIN
CREATE_MY_CONTAINER
HOLD_EXISTS.override
HOLD_ITEM_CHECKED_OUT.override
MR_HOLDS
OPAC_LOGIN
PERSISTENT_LOGIN
RENEW_CIRC
TITLE_HOLDS