Integrating Evergreen with Other Tools

Documentation Interest Group

Integrating Evergreen with Other Tools

Documentation Interest Group

Table of Contents

List of Tables

Part I. Introduction

Table of Contents

Chapter 1. About This Documentation

This guide was produced by the Evergreen Documentation Interest Group (DIG), consisting of numerous volunteers from many different organizations. The DIG has drawn together, edited, and supplemented pre-existing documentation contributed by libraries and consortia running Evergreen that were kind enough to release their documentation into the creative commons. Please see the Attributions section for a full list of authors and contributing organizations. Just like the software it describes, this guide is a work in progress, continually revised to meet the needs of its users, so if you find errors or omissions, please let us know, by contacting the DIG facilitators at docs@evergreen-ils.org.

This guide describes how to integrate Evergreen with other technologies, including Web browsers, Web sites, discovery layers, self-check machines, RFID equipment, SIP clients, auto-dialer phone scripts, and other applications.

Copies of this guide can be accessed in PDF and HTML formats from http://docs.evergreen-ils.org/.

Chapter 2. About Evergreen

Evergreen is an open source library automation software designed to meet the needs of the very smallest to the very largest libraries and consortia. Through its staff interface, it facilitates the management, cataloging, and circulation of library materials, and through its online public access interface it helps patrons find those materials.

The Evergreen software is freely licensed under the GNU General Public License, meaning that it is free to download, use, view, modify, and share. It has an active development and user community, as well as several companies offering migration, support, hosting, and development services.

The community’s development requirements state that Evergreen must be:

- Stable, even under extreme load.
- Robust, and capable of handling a high volume of transactions and simultaneous users.
- Flexible, to accommodate the varied needs of libraries.
- Secure, to protect our patrons’ privacy and data.
- User-friendly, to facilitate patron and staff use of the system.

Evergreen, which first launched in 2006 now powers over 544 libraries of every type – public, academic, special, school, and even tribal and home libraries – in over a dozen countries worldwide.

Part II. Adding Evergreen Search to Web Browsers

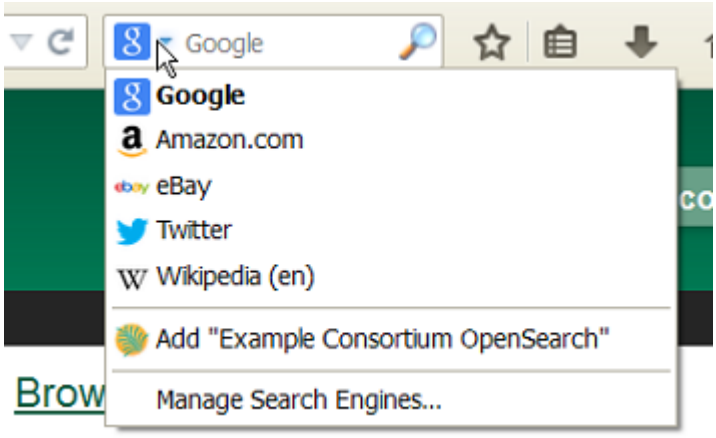
Table of Contents

Chapter 3. Adding OpenSearch to Firefox browser

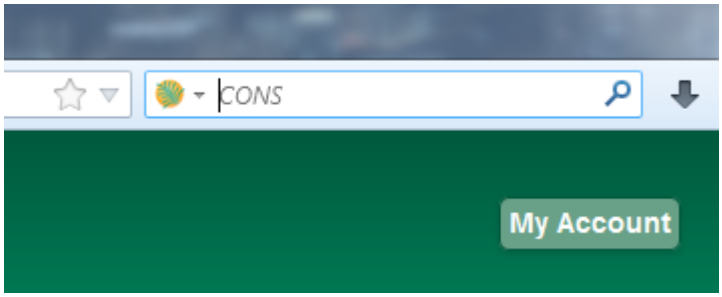
OpenSearch is a collection of simple formats for the sharing of search results. More information about OpenSearch can be found on their website.

The following example illustrates how to add an OpenSearch source to the list of search sources in a Firefox browser:

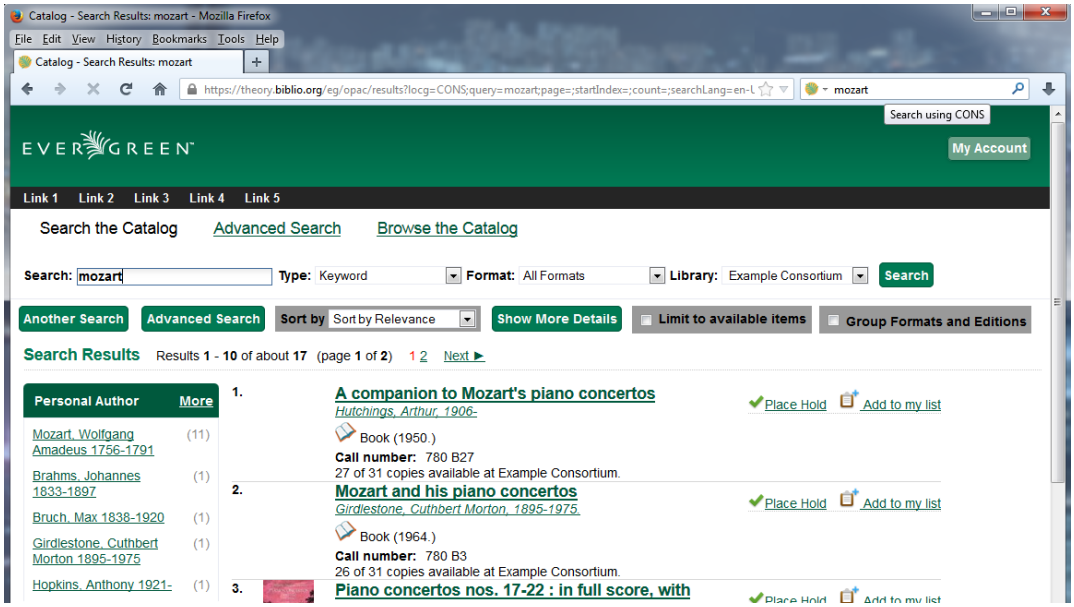
1. Navigate to any catalog page in your Firefox browser and click on the top right box's dropdown and select the option for Add "Example Consortium OpenSearch". The label will match the current scope.



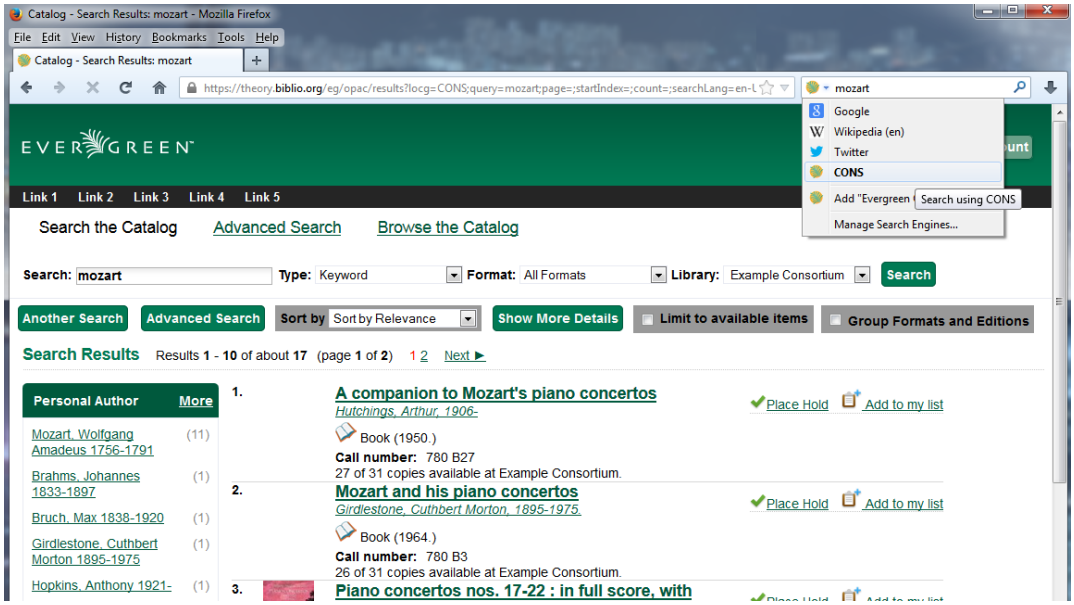
2. At this point, it will add a new search option for the location the catalog is currently using. In this example, that is CONS (searching the whole consortium).



3. Enter search terms to begin a keyword search using this source. The next image illustrates an example search for "mozart" using the sample bib record set.



4. You can select which search source to use by clicking on the dropdown picker.



Part III. Using Supercat

Table of Contents

Chapter 4. Introduction

You can use SuperCat to get data about ISBNs, metarecords, bibliographic records, and authority records.

Throughout this section, replace **<hostname>** with the domain or subdomain of your Evergreen installation to try these examples on your own system.

Chapter 5. ISBNs

Given one ISBN, Evergreen can return a list of related records and ISBNs, including alternate editions and translations. To use the Supercat oISBN tool, use http or https to access the following URL.

http://<hostname>/opac/extras/oisbn/<ISBN_to_query>

For example, the URL http://gapines.org/opac/extras/oisbn/0439136350 returns the following list of catalog record IDs and ISBNs:

```
<?xml version='1.0' encoding='UTF-8' ?>
<idlist metarecord='436139'>
  <isbn record='5652044'>9780606323475</isbn>
  <isbn record='5767568'>9780780673809</isbn>
  <isbn record='1350528'>9780807286029</isbn>
  <isbn record='5708164'>9780780669642</isbn>
  <isbn record='2372013'>043965548X</isbn>
  <isbn record='5804511'>8498386969</isbn>
  <isbn record='4132282'>9780786222742</isbn>
  <isbn record='1530458'>9788478885190</isbn>
  <isbn record='2003291'>0736650962</isbn>
  <isbn record='1993002'>8478885196</isbn>
  <isbn record='1187595'>9780439554923</isbn>
  <isbn record='4591175'>8478885196</isbn>
  <isbn record='5676282'>0807282324</isbn>
  <isbn record='2363352'>8478885196</isbn>
  <isbn record='2315122'>1480614998</isbn>
  <isbn record='2304130'>8478886559</isbn>
  <isbn record='2012565'>9780613371063</isbn>
  <isbn record='5763645'>9782070528189</isbn>
  <isbn record='2383286'>0786222743</isbn>
  <isbn record='2489670'>9780329232696</isbn>
  <isbn record='1681685'>9780807282311</isbn>
  <isbn record='2160095'>0807286028</isbn>
  <isbn record='2219885'>9789500421157</isbn>
  <isbn record='1934218'>9780613359580</isbn>
  <isbn record='5682871'>9781594130021</isbn>
  <isbn record='1281164'>0807283150</isbn>
  <isbn record='1666656'>0747542155</isbn>
  <isbn record='4717734'>8478886559</isbn>
</idlist>
```

Chapter 6. Records

Record formats

First, determine which format you'd like to receive data in. To see the available formats for bibliographic records, visit

http://<hostname>/opac/extras/supercat/formats/record

Similarly, authority record formats can be found at http://libcat.linnbenton.edu/opac/extras/supercat/formats/authority and metarecord formats can be found at http://libcat.linnbenton.edu/opac/extras/supercat/formats/metarecord

For example, http://gapines.org/opac/extras/supercat/formats/authority shows that the Georgia Pines catalog can return authority records in the formats opac, marc21, marc21-full, and marc21-uris. Supercat also includes the MIME type of each format, and sometimes also refers to the documentation for a particular format.

```
<?xml version='1.0' encoding='UTF-8' ?>
<formats>
  <format>
    <name>opac</name>
    <type>text/html</type>
  </format>
  <format>
    <name>marc21</name>
    <type>application/xml</type>
    <docs>http://www.loc.gov/marc/</docs>
  </format>
  <format>
    <name>marc21-full</name>
    <type>application/xml</type>
    <docs>http://www.loc.gov/marc/</docs>
  </format>
  <format>
    <name>marc21-uris</name>
    <type>application/xml</type>
    <docs>http://www.loc.gov/marc/</docs>
  </format>
</formats>
```

atom-full is currently the only format that includes holdings and availability data for a given bibliographic record.

Retrieve records

You can retrieve records using URLs in the following format:

http://<hostname>/opac/extras/supercat/retrieve/<format>/<record-type>/<record-ID>

For example, http://gapines.org/opac/extras/supercat/retrieve/mods/record/33333 returns the following record.

```
<?xml version="1.0"?>
<modsCollection xmlns="http://www.loc.gov/mods/" xmlns:mods="http://www.loc.gov/mods/" version="3.0">
```

```
 <mods xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:mods="http://www.loc.gov/mods/"
xsi:schemaLocation="http://www.loc.gov/mods/ http://www.loc.gov/standards/mods/mods.xsd">
   <titleInfo>
     <title>Words and pictures /</title>
   </titleInfo>
   <name type="personal">
     <namePart xmlns:xlink="http://www.w3.org/TR/xlink">Dodd, Siobhan</namePart>
     <role>
       <text>creator</text>
     </role>
   </name>
   <typeOfResource xmlns:xlink="http://www.w3.org/TR/xlink">text</typeOfResource>
   <originInfo xmlns:xlink="http://www.w3.org/TR/xlink">
     <place>
       <code authority="marc">mau</code>
     </place>
     <place>
       <text>Cambridge, Mass</text>
     </place>
     <publisher>Candlewick Press</publisher>
     <dateIssued>1992</dateIssued>
     <edition>1st U.S. ed.</edition>
     <issuance>monographic</issuance>
   </originInfo>
   <language authority="iso639-2b">eng</language>
   <physicalDescription>
     <form authority="marcform">print</form>
     <extent>1 v. (unpaged) : col. ill. ; 26 cm.</extent>
   </physicalDescription>
   <abstract>Simple text with picture cues accompany illustrations depicting scenes of everyday life familiar
to children, such as getting dressed, attending a party, playing in the park, and taking a bath.</abstract>
   <targetAudience>juvenile</targetAudience>
   <note type="statement of responsibility">Siobhan Dodds.</note>
   <subject xmlns:xlink="http://www.w3.org/TR/xlink" authority="lcshac">
     <topic>Family life</topic>
     <topic>Fiction</topic>
   </subject>
   <subject xmlns:xlink="http://www.w3.org/TR/xlink" authority="lcsh">
     <topic>Vocabulary</topic>
     <topic>Juvenile fiction</topic>
   </subject>
   <subject xmlns:xlink="http://www.w3.org/TR/xlink" authority="lcsh">
     <topic>Rebuses</topic>
   </subject>
   <subject xmlns:xlink="http://www.w3.org/TR/xlink" authority="lcsh">
     <topic>Picture puzzles</topic>
     <topic>Juvenile literature</topic>
   </subject>
   <subject xmlns:xlink="http://www.w3.org/TR/xlink" authority="lcsh">
     <topic>Picture books for children</topic>
   </subject>
   <subject xmlns:xlink="http://www.w3.org/TR/xlink" authority="lcsh">
     <topic>Picture dictionaries, English</topic>
     <topic>Juvenile literature</topic>
   </subject>
   <subject xmlns:xlink="http://www.w3.org/TR/xlink" authority="lcsh">
     <topic>Vocabulary</topic>
     <topic>Juvenile literature</topic>
   </subject>
   <classification authority="lcc">PZ7.D66275 Wo 1992</classification>
   <classification authority="lcc">PN6371.5 .D63 1992x</classification>
   <classification authority="ddc" edition="20">793.73</classification>
   <identifier type="isbn">1564020428 :</identifier>
   <identifier type="isbn">9781564020420</identifier>
   <identifier type="lccn">91071817</identifier>
   <recordInfo xmlns:xlink="http://www.w3.org/TR/xlink">
     <recordContentSource>DLC</recordContentSource>
```

```
      <recordCreationDate encoding="marc">920206</recordCreationDate>
      <recordChangeDate encoding="iso8601">20110608231047.0</recordChangeDate>
      <recordIdentifier source="GaAaGPL">33333</recordIdentifier>
    </recordInfo>
  </mods>
</modsCollection>
```

Recent records

SuperCat can return feeds of recently edited or created authority and bibliographic records:

http://<hostname>/opac/extras/feed/freshmeat/<feed-type>/<record-type>/<import-or-edit>/<limit>/<date>

Note the following features:

- The limit records imported or edited following the supplied date will be returned. If you do not supply a date, then the most recent limit records will be returned.
- If you do not supply a limit, then up to 10 records will be returned.
- feed-type can be one of atom, html, htmlholdings, marcxml, mods, mods3, or rss2.

Example: http://gapines.org/opac/extras/feed/freshmeat/atom/biblio/import/10/2008-01-01

Filtering by Org Unit

You can generate a similar list, with the added ability to limit by Org Unit, using the item-age browse axis.

To produce an RSS feed by item date rather than bib date, and to restrict it to a particular system within a consortium:

Example: http://gapines.org/opac/extras/browse/atom/item-age/ARL-BOG/1/10

Note the following:

- ARL-BOG should be the short name of the org unit you're interested in
- 1 is the page (since you are browsing through pages of results)
- 10 is the number of results to return per page

Modifying the atom portion of the URL to atom-full will include catalog links in the results:

Example: http://gapines.org/opac/extras/browse/atom-full/item-age/ARL-BOG/1/10

Modifying the atom portion of the URL to html-full will produce an HTML page that is minimally formatted:

Example: http://gapines.org/opac/extras/browse/html-full/item-age/ARL-BOG/1/10

Part IV. Using UnAPI

Table of Contents

Chapter 7. URL format

Evergreen's unAPI support includes access to many record types. For example, the following URL would fetch bib 267 in MODS32 along with holdings, volume, copy, and record attribute information:

https://example.org/opac/extras/unapi?id=tag::U2@bre/267{holdings_xml,acn,acp,mra}&format=mods32

To access the new unAPI features, the unAPI ID should have the following form:

- **tag::U2@**
- followed by class name, which may be
  - **bre** (bibs)
  - **biblio_record_entry_feed** (multiple bibs)
  - **acl** (copy locations)
  - **acn** (volumes)
  - **acnp** (call number prefixes)
  - **acns** (call number suffixes)
  - **acp** (copies)
  - **acpn** (copy notes)
  - **aou** (org units)
  - **ascecm** (copy stat cat entries)
  - **auri** (located URIs)
  - **bmp** (monographic parts)
  - **cbs** (bib sources)
  - **ccs** (copy statuses)
  - **circ** (loan checkout and due dates)
  - **holdings_xml** (holdings)
  - **mmr** (metarecords)
  - **mmr_holdings_xml** (metarecords with holdings)
  - **mmr_mra** (metarecords with record attributes)
  - **mra** (record attributes)

    - **sbsum** (serial basic summaries)
    - **sdist** (serial distributions)
    - **siss** (serial issues)
    - **sisum** (serial index summaries)
    - **sitem** (serial items)
    - **sssum** (serial supplement summaries)
    - **sstr** (serial streams)
    - **ssub** (serial subscriptions)
    - **sunit** (serial units)
- followed by **/**
- followed by a record identifier (or in the case of the **biblio_record_entry_feed** class, multiple IDs separated by commas)
- followed, optionally, by limit and offset in square brackets
- followed, optionally, by a comma-separated list of "includes" enclosed in curly brackets. The list of includes is the same as the list of classes with the following addition:
    - **bre.extern** (information from the non-MARC parts of a bib record)
- followed, optionally, by **/** and org unit; "-" signifies the top of the org unit tree
- followed, optionally, by **/** and org unit depth
- followed, optionally, by **/** and a path. If the path is **barcode** and the class is **acp**, the record ID is taken to be a copy barcode rather than a copy ID; for example, in **tag::U2@acp/ACQ140{acn,bre,mra}/-/0/barcode**, **ACQ140** is meant to be a copy barcode.
- followed, optionally, by **&format=** and the format in which the record should be retrieved. If this part is omitted, the list of available formats will be retrieved.

Part V. Phonelist.pm Module

Table of Contents

Chapter 8. Introduction

PhoneList.pm is a mod_perl module for Apache that works with Evergreen to generate callings lists for patron holds or overdues. It outputs a csv file that can be fed into an auto-dialer script to call patrons with little or no staff intervention. It is accessed and configured via a special URL and passing any parameters as a **Query String** on the URL. The parameters are listed in the table below.

Table 8.1. Parameters for the phonelist program:

user	Your Evergreen login. Typically your library's circ account. If you leave this off, you will be prompted to login.
passwd	The password for your Evergreen login. If you leave this off you will be prompted to login.
ws_ou	The ID of the system or branch you want to generate the list for (optional). If your account does not have the appropriate permissions for the location whose ID number you have entered, you will get an error.
skipemail	If present, skip patrons with email notification (optional).
addcount	Add a count of items on hold (optional). Only makes sense for holds.
overdue	Makes a list of patrons with overdues instead of holds. If an additional, numeric parameter is supplied, it will be used as the number of days overdue. If no such extra parameter is supplied, then the default of 14 days is used.

The URL is

https://your.evergreen-server.tld/phonelist

A couple of examples follow:

https://your.evergreen-server.tld/phonelist?
user=circuser&passwd=password&skipemail

The above example would sign in as user circuser with password of **password** and get a list of patrons with holds to call who do not have email notification turned on. It would run at whatever branch is normally associated with circuser.

https://your.evergreen-server.tld/phonelist?skipemail

The above example would do more or less the same, but you would be prompted by your browser for the user name and password.

If your browser or download script support it, you may also use conventional HTTP authentication parameters.

https://user:password@your.evergreen-server.tld/phonelist?overdue&ws_ou=2

The above logs in as **user** with **password** and runs overdues for location ID 2.

The following sections provide more information on getting what you want in your output.

Chapter 9. Adding Parameters

If you are not familiar with HTTP/URL query strings, the format is quite simple.

You add parameters to the end of the URL, the first parameter is separated from the URL page with a question mark (**?**) character. If the parameter is to be given an extra value, then that value follows the parameter name after an equals sign (**=**). Subsequent parameters are separated from the previous parameter by an ampersand (**&**).

Here is an example with 1 parameter that has no value:

https://your.evergreen-server.tld/phonelist?skipemail

An example of 1 argument with a value:

https://your.evergreen-server.tld/phonelist?overdue=21

An example of 2 arguments, 1 with a value and 1 without:

https://your.evergreen-server.tld/phonelist?overdue=21&skipemail

Any misspelled or parameters not listed in the table above will be ignored by the program.

Chapter 10. Output

On a successful run, the program will return a CSV file named phone.csv. Depending on your browser or settings you will alternately be prompted to open or save the file. Your browser may also automatically save the file in your Downloads or other designated folder. You should be able to open this CSV file in Excel, LibreOffice Base, any other spread sheet program, or a text editor.

If you have made a mistake and have mistyped your user name or password, or if you supply a ws_ou parameter with an ID where your user name does not have permission to look up holds or overdue information, then you will get an error returned in your browser.

Should your browser appear to do absolutely nothing at all. This is normal. When there is no information for you to download, the server will return a 200 NO CONTENT message to your browser. Most browsers respond to this message by doing nothing at all. It is possible for there to be no information for you to retrieve if you added the **`skipemail`** option and all of your notices for that day were sent via email, or if you ran this in the morning and then again in the afternoon and there was no new information to gather.

The program does indicate that it has already looked at a particular hold or overdue and will skip it on later runs. This prevents duplicates to the same patron in the same run. It will, however, create a **`duplicate`** for the same patron if a different copy is put on hold for that patron in between two runs.

The specific content of the CSV file will vary if you are looking at holds or overdues. The specific contents are described in the appropriate sections below.

Chapter 11. Holds

The **phonelist** program will return a list of patrons with copies on hold by default, so long as you do not use the **overdue** parameter. You may optionally get a number of items that patron currently has on hold by adding the **addcount** parameter.

As always, you can add the skipemail parameter to skip patrons with email notifications of their overdues, see Skipping patrons with email notification of holds as described below.

Table 11.1. Columns in the holds CSV file:

Name	Patron’s name first and last.
Phone	Patron’s phone number.
Barcode	Patron’s barcode.
Count	Number of copies on hold, if **addcount** parameter is used, otherwise this column is not present in the file.

Chapter 12. Overdues

If you add the **overdue** parameter, you can get a list of patrons with overdue copies instead of a list of patrons with copies on the hold shelf. By default, this will give you a list of patrons with copies that are 14 days overdue. If you'd like to specify a different number of days you can add the number after the parameter with an equals sign:

https://your.evergreen-server.tld/phonelist?overdue=21&ws_ou=2

The above will retrieve a list of patrons who have items that are 21 days overdue at the location with ID of 2.

The number of days is an exact lookup. This means that the program will look only at patrons who have items exactly 14 days or exactly the number of days specified overdue. It does not pull up any that are less than or greater than the number of days specified.

As always, you can add the skipemail parameter to skip patrons with email notifications of their overdues, see Skipping patrons with email notification of holds as described below.

Table 12.1. Columns in the overdues CSV file:

Name	Patron's name first and last.
Phone	Patron's phone number.
Barcode	Patron's barcode.
Titles	A colon-separated list of titles that the patron has overdue.

Chapter 13. Skipping patrons with email notification of holds

Skipping patrons who have email notification for their holds or overdues is very simple. You just need to add the **`skipemail`** parameter on the URL query string. Doing so will produce the list without the patrons who have email notification for overdues, or for all of their holds. Please note that if a patron has multiple holds available, and even one of these holds requests a phone-only notification, then that patron will still show on the list. For this option to exclude a patron from the holds list, the patron must request email notification on all of their current holds. In practice, we find that this is usually the case.

Chapter 14. Using the ws_ou parameter

Generally, you will not need to use the ws_ou parameter when using the phonelist program. The phonelist will look up the branch where your login account works and use that location when generating the list. However, if you are part of a multi-branch systems in a consortium, then the ws_ou parameter will be of interest to you. You can use it to specify which branch, or the whole system, you wish to search when running the program.

Chapter 15. Automating the download

If you’d like to automate the download of these files, you should be able to do so using any HTTP programming toolkit. Your client must accept cookies and follow any redirects in order to function.

Part VI. RFID Product Integration

Table of Contents

Chapter 16. Evergreen Integration with PV Supa GoodStuff RFID Reader

Introduction

This feature enables the Evergreen staff client to "talk" to the PV Supa Goodstuff RFID reader so that libraries can utilize PV Supa Goodstuff’s RFID technology when checking items in and out.

Administration

To use PV Supa Goodstuff, you must add code to the Admin module that Evergreen can use to identify the reader.

1. Click Admin → Workstation Admin → Server Add-ons.
2. Enter the code, pv_supa_goodstuff, to identify PV Supa Goodstuff in the Active Server Add-Ons field.
3. Click Update Active Add-Ons.
4. Look at the next field, Add-on Preferences. Enter information in the GoodStuff tab.
5. Check the Enabled check box to enable this add-on.
6. Enter the IP/Hostname of the hardware.
7. Enter the port.
8. Click Update.

Using RFID at the Circulation Desk

RFID functionality is available in the Circulation module via the check-out interface, the check-out function in the patron account, and the check-in interface.

From the Check-Out interface (F1):

1. Check the RFID checkbox if your library cards have embedded RFID chips that Evergreen can use to retrieve the patron barcode. RFID check boxes appear only if appropriate code words have been added in the Server Add-Ons.
2. Place the patron’s library card and/or item(s) on the PV Supa Goodstuff Reader. Evergreen will take you to the patron’s account. If item(s) with RFID chips have also been placed on the reader, and the corresponding checkbox is checked, then Evergreen will scan the item(s) into the patron’s account.

RFID check boxes are sticky, so if you have checked an RFID box once, then it will continue to be checked when you re-open the interface.

If you do not use RFID chips to retrieve patrons' accounts, then leave the RFID box unchecked. You can scan a patron barcode with a barcode scanner, and Evergreen will retrieve the patron data without using the RFID feature. From the patron’s account, Evergreen can check out items using the RFID reader. See the next section for details.

1. Click Done to complete the transaction, or close the window.

From the Check-Out tab in a patron’s record:

1. Open a patron’s record, and stack the item(s) to be checked out on the RFID reader. To retrieve item data using the RFID chips embedded in the item barcodes, click the RFID check box at the bottom of the patron account. When this box is checked, Evergreen will read the item(s) that are stacked on the RFID reader, check out the item(s), and disable the security bits.

2. Click Done to complete the transaction, or close the window.

Evergreen pop-up messages, such as an Alert Message or Item Already Circulating may appear during transactions. Two new dialogs specific to PV Supa Goodstuff may also appear. One dialog, Incomplete Sets, allows you to continue checking out an incomplete set, such as a CD collection, or you can ask the hardware to rescan the RFID tags to look again for a full set. The second dialog allows you to manually attempt to set or unset the security bit on an item if the automatic attempt failed.

From the Check In interface:

1. Click the RFID check box.

2. Place the items on the PV Supa Goodstuff Reader.

3. Evergreen will tell the reader to check in the item(s) and enable the security bits. The item(s) appear in a list on the screen.

Part VII. Adding an Evergreen search form to a web page

Table of Contents

Chapter 17. Introduction

To enable users to quickly search your Evergreen catalog, you can add a simple search form to any HTML page. The following code demonstrates how to create a quick search box suitable for the header of your web site:

Chapter 18. Simple search form

```
<form action="http://example.com/eg/opac/results" method="get" accept-charset="UTF-8">    <!-- ❶ -->
    <input type="search" alt="Catalog Search" maxlength="200"
            size="20" name="query"
            placeholder="Search catalog for..." />
    <input type="hidden" name="qtype" value="keyword" />           <!-- ❷ -->
    <input type="hidden" name="locg" value="4" />                  <!-- ❸ -->
    <input type="submit" value="Search" />
</form>
```

❶ Replace 'example.com' with the hostname for your catalog. To link to the Kid’s OPAC instead of the TPAC, replace 'opac' with 'kpac'.

❷ Replace 'keyword' with 'title', 'author', 'subject', or 'series' if you want to provide more specific searches. You can even specify 'identifier|isbn' for an ISBN search.

❸ Replace '4' with the ID number of the organizational unit at which you wish to anchor your search. This is the value of the 'locg' parameter in your normal search.

Chapter 19. Advanced search form

```
<form role="search" id="searchForm" method="get" class="searchform" action="http://your_catalog/eg/opac/results"
 accept-charset="UTF-8">
     <label id="searchLabel" for="search"><b>Search the Catalog: </b></label>
     <input type="search" value="" name="query" id="search" size="30">
     <label id="search_qtype_label"><b>Type:</b>
     <select name="qtype" id="qtype"    aria-label="Select query type:">
        <option value='keyword' selected="selected">Keyword</option>
        <option value='title'>Title</option>
        <option value='jtitle'>Journal Title</option>
        <option value='author'>Author</option>
        <option value='subject'>Subject</option>
        <option value='series'>Series</option>
     </select>
     </label>    

     <label id="search_itype_label"><b>Format: </b>
     <select id='item_type_selector' name='fi:item_type' aria-label="Select item type:">
       <option value=''>All Formats</option>
       <option value='a'>Books and Journals</option>
       <option value='i'>Nonmusical Sound Recording</option>
       <option value='j'>Musical Sound Recording</option>
       <option value='g'>Video</option>
       </select>    
     </label>

    <label id="search_locg_label"><b>Library: </b>
    <select aria-label='Select search library'   name='locg'>
            <option value='1' class="org_unit">
                All Libraries
            </option>
            <option value='2' selected="selected" class="org_unit">
                 Central Library
            </option>
             <option value='10'   class="org_unit">
                    Little Library
            </option>
     </select>
     </label>
          <input class="searchbutton" type="submit" value="SEarch" />
</form>
```

Chapter 20. Encoding

For non English characters it is vital to set the attribute **accept-charset="UTF-8"** in the form tag (as in the examples above). If the parameter is not set, records with non English characters will not be retrieved.

Chapter 21. Setting the document type

You can set the document types to be searched using the attribute **option value=** in the form. For the value use MARC 21 code defining the type of record (i.e. Leader, position 06).

For example, for musical recordings you could use **<option value='j'>Musical Sound Recording</option>**

Chapter 22. Setting the library

Instead of searching the entire consortium, you can set the Library to be searched in using the attribute **`option value=`** in the form. For the value use Evergreen database.organization unit ID.

Part VIII. SIP Server

Table of Contents

Chapter 23. About the SIP Protocol

SIP, standing for **Standard Interchange Protocol**, was developed by the **3M corporation** to be a common protocol for data transfer between ILS' (referred to in **SIP** as an ACS, or Automated Circulation System) and a third party device. Originally, the protocol was developed for use with 3M SelfCheck (often abbreviated SC, not to be confused with Staff Client) systems, but has since expanded to other companies and devices. It is now common to find **SIP** in use in several other vendors' SelfCheck systems, as well as other non-SelfCheck devices. Some examples include:

- Patron Authentication (computer access, subscription databases)
- Automated Material Handling (AMH)
    - The automated sorting of items, often to bins or book carts, based on shelving location or other programmable criteria

Chapter 24. Installing the SIP Server

This is a rough intro to installing the **SIP** server for Evergreen.

Getting the code

Current **SIP** server code lives at in the Evergreen git repository:

```
cd /opt
git clone git://git.evergreen-ils.org/SIPServer.git SIPServer
```

Configuring the Server

1. Type the following commands from the command prompt:

   ```
   $ sudo su opensrf
   $ cd /openils/conf
   $ cp oils_sip.xml.example oils_sip.xml
   ```

2. Edit oils_sip.xml. Change the commented out <server-params> section to this:

   ```
   <server-params
      min_spare_servers='1'
      max_spare_servers='2'
      min_servers='3'
      max_servers='25'
                />
   ```

3. max_servers will directly correspond to the number of allowed **SIP** clients. Set the number accordingly, but bear in mind that too many connections can exhaust memory. On a 4G RAM/4 CPU server (that is also running evergreen), it is not recommended to exceed 100 **SIP** client connections.

Setting the encoding

SIPServer looks for the encoding in the following places:

1. An **encoding** attribute on the **account** element for the currently active SIP account.

2. The **encoding** element that is a child of the **institution** element of the currently active SIP account.

3. The **encoding** element that is a child of the **implementation_config** element that is itself a child of the **institution** element of the currently active SIP account.

4. If none of the above exist, then the default encoding (ASCII) is used.

Option 3 is a legacy option. It is recommended that you alter your configuration to move this element out of the **implementation_config** element and into its parent **institution** element.

Ideally, SIPServer should not look into the implementation config, and this check may be removed at some time in the future.

Datatypes

The **msg64_hold_datatype** setting is similar to **msg64_summary_datatype**, but affects holds instead of circulations. When set to **barcode**, holds information will be delivered as a set of copy barcodes instead of title strings for patron info requests. With barcodes, SIP clients can both find the title strings for display (via item info requests) and make subsequent hold-related action requests, like holds cancellation.

Adding SIP Users

1. Type the following commands from the command prompt:

```
$ sudo su opensrf
$ cd /openils/conf
```

2. In the **<accounts>** section, add **SIP** client login information. Make sure that all **<logins>** use the same institution attribute, and make sure the institution is listed in **<institutions>**. All attributes in the **<login>** section will be used by the **SIP** client.

3. In Evergreen, create a new profile group called **SIP**. This group should be a sub-group of **Users** (not **Staff** or **Patrons**). Set Editing Permission as group_application.user.sip_client and give the group the following permissions:

```
COPY_CHECKIN
COPY_CHECKOUT
CREATE_PAYMENT
RENEW_CIRC
VIEW_CIRCULATIONS
VIEW_COPY_CHECKOUT_HISTORY
VIEW_PERMIT_CHECKOUT
VIEW_USER
VIEW_USER_FINES_SUMMARY
VIEW_USER_TRANSACTIONS
```

   OR use SQL like:

```
INSERT INTO permission.grp_tree (name,parent,description,application_perm)
VALUES ('SIP', 1, 'SIP2 Client Systems', 'group_application.user.sip_client');

INSERT INTO
  permission.grp_perm_map (grp, perm, depth, grantable)
SELECT
  g.id, p.id, 0, FALSE
FROM
  permission.grp_tree g,
  permission.perm_list p
WHERE
  g.name = 'SIP' AND
  p.code IN (
    'COPY_CHECKIN',
    'COPY_CHECKOUT',
    'RENEW_CIRC',
    'VIEW_CIRCULATIONS',
```

```
    'VIEW_COPY_CHECKOUT_HISTORY',
    'VIEW_PERMIT_CHECKOUT',
    'VIEW_USER',
    'VIEW_USER_FINES_SUMMARY',
    'VIEW_USER_TRANSACTIONS'
 );
```

Verify:

```
SELECT *
FROM permission.grp_perm_map pgpm
    INNER JOIN permission.perm_list ppl ON pgpm.perm = ppl.id
    INNER JOIN permission.grp_tree pgt ON pgt.id = pgpm.grp
WHERE pgt.name = 'SIP';
```

4. For each account created in the **<login>** section of oils_sip.xml, create a user (via the staff client user editor) that has the same username and password and put that user into the **SIP** group.

The expiration date will affect the **SIP** users' connection so you might want to make a note of this somewhere.

Running the server

To start the **SIP** server type the following commands from the command prompt:

```
$ sudo su opensrf
```

```
$ oils_ctl.sh -a [start|stop|restart]_sip
```

Logging-SIP

Syslog

It is useful to log **SIP** requests to a separate file especially during initial setup by modifying your syslog config file.

1. Edit syslog.conf.

```
$ sudo vi /etc/syslog.conf  # maybe /etc/rsyslog.conf
```

2. Add this:

```
local6.*                -/var/log/SIP_evergreen.log
```

3. Syslog expects the logfile to exist so create the file.

```
$ sudo touch /var/log/SIP_evergreen.log
```

4. Restart sysklogd.

```
$ sudo /etc/init.d/sysklogd restart
```

Syslog-NG

1. Edit logging config.

```
sudo vi /etc/syslog-ng/syslog-ng.conf
```

2. Add:

```
# +SIP2+ for Evergreen
filter    f_eg_sip { level(warn, err, crit) and facility(local6); };
destination eg_sip { file("var/log/SIP_evergreen.log"); };
log { source(s_all); filter(f_eg_sip); destination(eg_sip); };
```

3. Syslog-ng expects the logfile to exist so create the file.

```
$ sudo touch /var/log/SIP_evergreen.log
```

4. Restart syslog-ng

```
$ sudo /etc/init.d/syslog-ng restart
```

Testing Your SIP Connection

- In the root directory of the SIPServer code:

```
$ cd SIPServer/t
```

- Edit SIPtest.pm, change the $instid, $server, $username, and $password variables. This will be enough to test connectivity. To run all tests, you'll need to change all the variables in the Configuration section.

```
$ PERL5LIB=../ perl 00sc_status.t
```

  This should produce something like:

```
1..4
ok 1 - Invalid username
ok 2 - Invalid username
ok 3 - login
ok 4 - SC status
```

- Don't be dismayed at Invalid Username. That's just one of the many tests that are run.

More Testing

Once you have opened up either the **SIP** OR **SIP2** ports to be accessible from outside you can do some testing via **telnet**. In the following tests:

- Replace **$server** with your server hostname (or **localhost** if you want to skip testing external access for now);

- Replace **$username**, **$password**, and **$instid** with the corresponding values in the **<accounts>** section of your SIP configuration file;
- Replace the **$user_barcode** and **$user_password** variables with the values for a valid user.
- Replace the **$item_barcode** variable with the values for a valid item.

1. Start by testing your ability to log into the SIP server:

We are using 6001 here which is associated with **SIP2** as per our configuration.

```
$ telnet $server 6001
Connected to $server.
Escape character is '^]'.
9300CN$username|CO$password|CP$instid
```

If successful, the SIP server returns a **941** result. A result of **940**, however, indicates an unsuccessful login attempt. Check the **<accounts>** section of your SIP configuration and try agin.

2. Once you have logged in successfully, replace the variables in the following line and paste it into the telnet session:

```
2300120080623    172148AO$instid|AA$user_barcode|AC$password|AD$user_password
```

If successful, the SIP server returns the patron information for $user_barcode, similar to the following:

```
24  Y           00120100113    170738AEFirstName MiddleName LastName|AA$user_barcode|BLY|CQY
|BHUSD|BV0.00|AFOK|AO$instid|
```

The response declares it is a valid patron BLY with a valid password CQY and shows the user's **$name**.

3. To test the SIP server's item information response, issue the following request:

```
1700120080623    172148AO$instid|AB$item_barcode|AC$password
```

If successful, the SIP server returns the item information for $item_barcode, similar to the following:

```
1803020120160923    190132AB30007003601852|AJRégion de Kamouraska|CK001|AQOSUL|APOSUL|BHCAD
|BV0.00|BGOSUL|CSCA2 PQ NR46 73R
```

The response declares it is a valid item, with the title, owning library, permanent and current locations, and call number.

Chapter 25. SIP Communication

SIP generally communicates over a **TCP** connection (either raw sockets or over **telnet**), but can also communicate via serial connections and other methods. In Evergreen, the most common deployment is a **RAW** socket connection on port 6001.

SIP communication consists of strings of messages, each message request and response begin with a 2-digit “command” - Requests usually being an odd number and responses usually increased by 1 to be an even number. The combination numbers for the request command and response is often referred to as a Message Pair (for example, a 23 command is a request for patron status, a 24 response is a patron status, and the message pair 23/24 is patron status message pair). The table in the next section shows the message pairs and a description of them.

For clarification, the “Request” is from the device (selfcheck or otherwise) to the ILS/ACS. The response is... the response to the request ;).

Within each request and response, a number of fields (either a fixed width or separated with a | [pipe symbol] and preceded with a 2-character field identifier) are used. The fields vary between message pairs.

Pair	Name	Supported?	Details
01	Block Patron	Yes	01/Block_Patron - ACS responds with 24 Patron Status Response
09-10	Checkin	Yes (with extensions)	09/10_Checkin
11-12	Checkout	Yes (no renewals)	11/12_Checkout
15-16	Hold	Partially supported	15/16_Hold
17-18	Item Information	Yes (no extensions)	17/18_Item_Information
19-20	Item Status Update	No	19/20_Item_Status_Update - Returns Patron Enable response, but doesn’t make any changes in EG
23-24	Patron Status	Yes	23/24_Patron_Status - 63/64 “Patron Information” preferred
25-26	Patron Enable	No	25/26_Patron_Enable - Used during system testing and validation
29-30	Renew	Yes	29/30_Renew
35-36	End Session	Yes	35/36_End_Session
37-38	Fee Paid	Yes	37/38_Fee_Paid
63-64	Patron Information	Yes (no extensions)	63/64_Patron_Information
65-66	Renew All	Yes	65/66_Renew_All
93-94	Login	Yes	93/94_Login - Must be first command to

			Evergreen ACS (via socket) or **SIP** will terminate
97-96	Resend last message	Yes	97/96_Resend
99-98	SC-ACS Status	Yes	99/98_SC_and_ACS_Status

01 Block Patron

A selfcheck will issue a Block Patron command if a patron leaves their card in a selfcheck machine or if the selfcheck detects tampering (such as attempts to disable multiple items during a single item checkout, multiple failed pin entries, etc).

In Evergreen, this command does the following:

- User alert message: CARD BLOCKED BY SELF-CHECK MACHINE (this is independent of the AL Blocked Card Message field).
- Card is marked inactive.

The request looks like:

```
01<card retained><date>[fields AO, AL, AA, AC]
```

Card Retained: A single character field of Y or N - tells the ACS whether the SC has retained the card (ex: left in the machine) or not.

Date: An 18 character field for the date/time when the block occurred.

Format: YYYYMMDDZZZZHHMMSS (ZZZZ being zone - 4 blanks when local time, “Z” (3 blanks and a Z) represents UTC(GMT/Zulu)

Fields: See Fields for more details.

The response is a 24 “Patron Status Response” with the following:

- Charge privileges denied
- Renewal privileges denied
- Recall privileges denied (hard-coded in every 24 or 64 response)
- hold privileges denied
- Screen Message 1 (AF): blocked
- Patron

09/10 Checkin

~The request looks like:

```
09<No block (Offline)><xact date><return date>[Fields AP,AO,AB,AC,CH,BI]
```

No Block (Offline): A single character field of Y or N - Offline transactions are not currently supported so send N.

xact date: an 18 character field for the date/time when the checkin occurred. Format: YYYYMMDDZZZZHHMMSS (ZZZZ being zone - 4 blanks when local time, “Z” (3 blanks and a Z) represents UTC(GMT/Zulu)

Fields: See Fields for more details.

The response is a 10 “Checkin Response” with the following:

```
10<resensitize><magnetic media><alert><xact date>[Fields AO,AB,AQ,AJ,CL,AA,CK,CH,CR,CS,CT,CV,CY,DA,AF,AG]
```

Example (with a remote hold):

```
09N20100507    16593720100507    165937APCheckin Bin 5|AOBR1|AB1565921879|ACsip_01|

101YNY20100623    165731AOBR1|AB1565921879|AQBR1|AJPerl 5 desktop reference|CK001|CSQA76.73.P33V76 1996
|CTBR3|CY373827|DANicholas Richard Woodard|CV02|
```

Here you can see a hold alert for patron CY 373827, named DA Nicholas Richard Woodard, to be picked up at CT “BR3”. Since the transaction is happening at AO “BR1”, the alert type CV is 02 for hold at remote library. The possible values for CV are:

- 00: unknown
- 01: local hold
- 02: remote hold
- 03: ILL transfer (not used by EG)
- 04: transfer
- 99: other

The logic for Evergreen to determine whether the content is magnetic_media comes from or search_config_circ_modifier. The default is non-magnetic. The same is true for media_type (default 001). Evergreen does not populate the collection_code because it does not really have any, but it will provide the call_number where available.

Unlike the **item_id** (barcode), the **title_id** is actually a title string, unless the configuration forces the return of the bib ID.

Don’t be confused by the different branches that can show up in the same response line.

- AO is where the transaction took place,
- AQ is the “permanent location”, and
- CT is the destination location (i.e., pickup lib for a hold or target lib for a transfer).

11/12 Checkout

15/16 Hold

Evergreen supports the Hold message for the purpose of canceling holds. It does not currently support creating hold requests via SIP2.

17/18 Item Information

The request looks like:

```
17<xact_date>[fields: AO,AB,AC]
```

The request is very terse. AC is optional.

The following response structure is for **SIP2**. (Version 1 of the protocol had only 6 total fields.)

```
18<circulation_status><security_marker><fee_type><xact_date>
[fields: CF,AH,CJ,CM,AB,AJ,BG,BH,BV,CK,AQ,AP,CH,AF,AG,+CT,+CS]
```

Example:

```
1720060110    215612AOBR1|ABno_such_barcode|

1801010120100609    162510ABno_such_barcode|AJ|

1720060110    215612AOBR1|AB1565921879|

1810020120100623    171415AB1565921879|AJPerl 5 desktop reference|CK001|AQBR1|APBR1|BGBR1
|CTBR3|CSQA76.73.P33V76 1996|
```

The first case is with a bogus barcode. The latter shows an item with a circulation_status of 10 for in transit between libraries. The known values of **`circulation_status`** are enumerated in the spec.

EXTENSIONS: The CT field for destination location and CS call number are used by Automated Material Handling systems.

19/20 Item Status Update

23/24 Patron Status

Example:

```
2300120060101    084235AOUWOLS|AAbad_barcode|ACsip_01|ADbad_password|
24YYYY          00120100507    013934AE|AAbad_barcode|BLN|AOUWOLS|
2300120060101    084235AOCONS|AA999999|ACsip_01|ADbad_password|
24  Y           00120100507    022318AEDoug Fiander|AA999999|BLY|CQN|BHUSD|BV0.00|AFOK|AOCONS|
2300120060101    084235AOCONS|AA999999|ACsip_01|ADuserpassword|LY|CQN|BHUSD|BV0.00|AFOK|AOCONS|
24  Y           00120100507    022803AEDoug Fiander|AA999999|BLY|CQY|BHUSD|BV0.00|AFOK|AOCONS|
```

1. The BL field (**SIP2**, optional) is valid patron, so the N value means bad_barcode doesn't match a patron, the Y value means 999999 does.
2. The CQ field (**SIP2**, optional) is valid password, so the N value means bad_password doesn't match 999999's password, the Y means userpassword does.

So if you were building the most basic **SIP2** authentication client, you would check for |CQY| in the response to know the user's barcode and password are correct (|CQY| implies |BLY|, since you cannot check the password unless the barcode exists). However, in practice, depending on the application, there are other factors to consider in authentication, like whether the user is blocked from checkout, owes excessive fines, reported their card lost, etc. These limitations are reflected in the 14-character patron status string immediately following the 24 code. See the field definitions in your copy of the spec.

25/26 Patron Enable

Not yet supported.

29/30 Renew

Evergreen supports the Renew message. Evergreen checks whether a penalty is specifically configured to block renewals before blocking any SIP renewal.

35/36 End Session

```
3520100505    115901AOBR1|AA999999|
```

```
36Y20100507    161213AOCONS|AA999999|AFThank you!|
```

The Y/N code immediately after the 36 indicates success/failure. Failure is not particularly meaningful or important in this context, and for evergreen it is hardcoded Y.

37/38 Fee Paid

Evergreen supports the Fee Paid message.

63/64 Patron Information

Attempting to retrieve patron info with a bad barcode:

```
6300020060329    201700          AOBR1|AAbad_barcode|

64YYYY          00020100623    141130000000000000000000000000AE|AAbad_barcode|BLN|AOBR1|
```

Attempting to retrieve patron info with a good barcode (but bad patron password):

```
6300020060329    201700          AOBR1|AA999999|ADbadpwd|

64  Y           00020100623    141130000000000000000000000000AA999999|AEDavid J. Fiander|BHUSD|BV0.00
|BD2 Meadowvale Dr. St Thomas, ON Canada

90210|BEdjfiander@somemail.com|BF(519) 555 1234|AQBR1|BLY|CQN|PB19640925|PCPatrons
|PIUnfiltered|AFOK|AOBR1|
```

See 23/24 Patron Status for info on **BL** and **CQ** fields.

65/66 Renew All

Evergreen supports the Renew All message.

93/94 Login

Example:

```
9300CNsip_01|CObad_value|CPBR1|

[Connection closed by foreign host.]
...

9300CNsip_01|COsip_01|CPBR1|

941
```

941 means successful terminal login. 940 or getting dropped means failure.

When using a version of SIPServer that supports the feature, the Location (CP) field of the Login (93) message will be used as the workstation name if supplied. Blank or missing location fields will be ignored. This allows users or reports to determine which selfcheck performed a circulation.

97/96 Resend

99/98 SC and ACS Status

```
99<status code><max print width><protocol version>
```

All 3 fields are required:

- 0: SC is OK
- 1: SC is out of paper
- 2: SC shutting down
- status code - 1 character
- max print width - 3 characters - the integer number of characters the client can print
- protocol version - 4 characters - x.xx

```
98<on-line status><checkin ok><checkout ok><ACS renewal policy>
<status update ok><offline ok><timeout period>

<retries allowed><date/time sync><protocol version><institution id>
<library name><supported messages><terminal

location><screen message><print line>
```

Example:

```
9910302.00

98YYYYNN60000320100510    1717202.00AOCONS|BXYYYYYYYYYYNYNNNYN|
```

The Supported Messages field **BX** appears only in **SIP2**, and specifies whether 16 different **SIP** commands are supported by the **ACS** or not.

Fields

All fixed-length fields in a communication will appear before the first variable-length field. This allows for simple parsing. Variable-length fields are by definition delimited, though there will not necessarily be an initial delimiter between the last fixed-length field and the first variable-length one. It would be unnecessary, since you should know the exact position where that field begins already.

Chapter 26. Patron privacy and the SIP protocol

SIP traffic includes a lot of patron information, and is not encrypted by default. It is strongly recommended that you encrypt any SIP traffic.

SIP server configuration

On the SIP server, use **iptables** or **etc/hosts** to allow SSH connections on port 22 from the SIP client machine. You will probably want to have very restrictive rules on which IP addresses can connect to this server.

SSH tunnels on SIP clients

SSH tunnels are a good fit for use cases like self-check machines, because it is relatively easy to automatically open the connection. Using a VPN is another option, but many VPN clients require manual steps to open the VPN connection.

1. If the SIP client will be on a Windows machine, install cygwin on the SIP client.
2. On the SIP client, use **ssh-keygen** to generate an SSH key.
3. Add the public key to /home/my_sip_user/.ssh/authorized_keys on your SIP server to enable logins without using the UNIX password.
4. Configure an SSH tunnel to open before every connection. You can do this in several ways:
   a. If the SIP client software allows you to run an arbitrary command before each SIP connection, use something like this:

      ```
      ssh -f -L 6001:localhost:6001 my_sip_user@my_sip_server.com sleep 10
      ```

   b. If you feel confident that the connection won't get interrupted, you can have something like this run at startup:

      ```
      ssh -f -N -L 6001:localhost:6001 my_sip_user@my_sip_server.com
      ```

   c. If you want to constantly poll to make sure that the connection is still running, you can do something like this as a cron job or scheduled task on the SIP client machine:

```
#!/bin/bash
instances=`/bin/ps -ef | /bin/grep ssh | /bin/grep -v grep | /bin/wc -l`
if [ $instances -eq 0 ]; then
  echo "Restarting ssh tunnel"
  /usr/bin/ssh -L 6001:localhost:6001 my_sip_user@my_sip_server.com -f -N
fi
```

Appendix A. Attributions

Current DIG Members

- Hilary Caws-Elwitt, Susquehanna County Library
- Karen Collier, Kent County Public Library
- George Duimovich, NRCan Library
- Lynn Floyd, Anderson County Library
- Sally Fortin, Equinox Software
- Wolf Halton, Lyrasis
- Jennifer Pringle, SITKA
- June Rayner, eiNetwork
- Steve Sheppard
- Ben Shum, Bibliomation
- Roni Shwaish, eiNetwork
- Robert Soulliere, Mohawk College
- Remington Steed, Calvin College

- Tim Spindler, C/W MARS
- Jane Sandberg, Linn-Benton Community College
- Lindsay Stratton, Pioneer Library System
- Yamil Suarez, Berklee College of Music
- Jenny Turner, PALS

Appendix B. Admonitions

- Note

- warning

- caution

- tip

Appendix C. Licensing

Index