# Learning *Online* Network with CAPA

*Domain Coordination Tutorial And Manual*

May 29, 2015

LON-CAPA Group

Michigan State University

# Contents

**1 Introduction** **3**

**2 Domain Configuration** **4**
2.1 Log-in Screen Customization . . . . . 4
2.2 Setting E-mail Addresses for Administrators and Support . . . . . 5
2.3 Default Color Schemes . . . . . 6
2.4 Setting Domain Defaults: Authentication, Language, Time Zone, Portal and User Types . . . . . 7
2.5 Setting Domain Defaults: Blogs, Portfolios, Personal Information Pages, WebDAV, and Quotas . . . . . 8
2.6 Identity Management: Searching for Users in an Institutional Directory . . . 9
2.7 Identity Management: Creating New Users . . . . . 10
2.8 Identity Management: Modifying Existing Users . . . . . 12
2.9 Identity Management: Automated Updates of User Information . . . . . 12
2.10 Automated Course Creation . . . . . 13
2.11 Course/Community Requests . . . . . 13
2.12 Course Catalogs . . . . . 15
2.13 Automated Enrollment in Official Courses . . . . . 16
2.14 Course/Community Defaults . . . . . 18
2.15 Course/Community Self-enrollment . . . . . 19
2.16 Authoring Space Requests . . . . . 19
2.17 Bubblesheet Data Formats . . . . . 20
2.18 Access to Server Status Pages . . . . . 22
2.19 User Session Hosting/Offloading . . . . . 24
2.20 Load Balancing with Dedicated LON-CAPA Server . . . . . 25
2.21 Encrypting server traffic with SSL . . . . . 27

**3 Domain Management** **31**
3.1 Creating Domain Coordinators . . . . . 31
3.2 Revoking Domain Coordinator Roles . . . . . 31
3.3 Scheduling of Scripts Run Periodically . . . . . 32
3.4 Creating Courses . . . . . 33

**4 Integration with Institutional Systems** **34**
4.1 Institutional Authentication (non-SSO) . . . . . 34
4.2 Shibboleth Authentication (SSO) . . . . . 36
4.3 CAS Authentication (SSO) . . . . . 43
4.4 Institutional User Categories/Affiliations . . . . . 44
4.5 Format Rule Definitions and Checks: Usernames and IDs . . . . . 46
4.6 Institutional Directory Information . . . . . 50
4.7 Search Filters for Official Course Categories . . . . . 59
4.8 Validation of Requests for Official Courses . . . . . 61

# 1 Introduction

The Domain Coordination Manual includes both a description of tasks which require standard LON-CAPA interaction via a web browser, with the Domain Coordinator role active, but also tasks which require command line access to the primary library server in a domain. In some cases one individual may complete both these types of task, whereas in others a separate Systems Administrator may need to be called upon for tasks completed from the command line.

With the Domain Coordinator role active, the Main Menu provides a Domain Coordinator with access to the functionality needed to complete the routine tasks which a Domain Coordinator carries out, including creation of courses, approval of course requests, and assignment of author roles. Courses can be created interactively by completing a web form or can be created in batch mode by uploading a file containing course specifications for one or more courses. The right to submit course requests via a web form may also be granted to groups of users, and requests can be set to be processed automatically, or require Domain Coordinator approval. Addition of users/roles can similarly be carried out interactively, or by uploading a text file of users. The Domain Coordinator may also periodically need to modify domain settings via the Domain Configuration menu.

LON-CAPA provides hooks to permit integration with institutional information systems to support the following procedures:

- Authentication via an institutional authentication service (e.g., LDAP)
- Automatic updates of classlists
- Automatic updates of user information
- Automatic processing of validated course requests for "official" courses.
- Retrieval of institutional user information for individual users
- Searches for users at the institution
- Automatic import of student photos from an institutional repository

There are three perl modules included in LON-CAPA (*localauth.pm localenroll.pm*, and *localstudentphoto.pm* all located in */home/httpd/lib/perl*) which need to be customized to provide integration with institutional systems.

Although customization and testing will involve a systems administrator, and programmer(s) at an institution, a Domain Coordinator is likely to be involved in the design and testing phases of the integration, if not in the actual implementation. Set-up of a standalone LON-CAPA instance on a separate server is advised for the purposes of implementing and testing institutional integration, before enabling the new functionality on the production system.

Besides the integration described above, customization of *localenroll.pm* is also needed (at present) to define the titles used for official course categories (e.g., Year, Semester, Department, Number) which can be used as filters when searching the Course/Community Catalog.

LON-CAPA installation and/or update will provide unmodifed copies of the three customizable files in */home/httpd/lib/perl*, each with -std appended, i.e., *localauth-std.pm*, *localenroll-std.pm*, and *localstudentphoto-std.pm*. These files receive updates when ./UPDATE is run to update a LON-CAPA installation to a newer version, while their customized equivalents will be left untouched.

If you have previously customized localenroll.pm it is recommended that you compare the contents of localenroll.pm and localenroll-std.pm after an update to see if there are new subroutines (which exist as stubs in localenroll-std.pm) which can be copied to your custom localenroll.pm and later customized, should you wish to use that functionality.

# 2 Domain Configuration

## 2.1 Log-in Screen Customization

If your domain has more than one server you have the option to configure whether any of the servers will redirect to another server whenever the log-in page is requested. This can be useful if you maintain a portal or "Load Balancer" server which forms your institution's gateway to LON-CAPA. You can specify the path to which the user should be redirected, and also whether log-in page requests from specific IP addresses should be exempt from the redirection. The exemption is useful if you run a monitoring script which tests log-in, course display, and logout periodically for each of your LON-CAPA servers.

If your domain only has one LON-CAPA server, or you have multiple servers and will display their log-in pages, their appearance can be customized as follows:

- uploading custom image files
- changing colors of text, links or backgrounds
- enabling/disabling display of specific links

Logos displayed in the login page configuration table are scaled down from the full size used in the login-page itself.

The following elements are configurable:

- Header image at the top of the page
- Main Logo centered in the upper part of the main panel
- Domain logo in the lower left corner of the main panel
- Header above the login panel - can also be set to use text ("Log in") instead of an image.
- Background colors for the page itself, the main panel, and the left (side) panel.
- Text color used for text on the page

- Text colors used for active, visited and unvisited links
- Enable/disable display of three links:
  - Course/Community Catalog, for a catalog of courses and communities
  - Admin E-mail, for the e-mail address of the administrator
  - Contact Helpdesk, to display a web form used to submit a help request
  - New User, for users to create their own accounts
- Default colors for links in the page, depending on status: either active, visited or default (if neither apply).

A "Log-in Help" link will be displayed immediately above any of the three optional links: Catalog, Contact Helpdesk and New User. Configuration options determine to which file(s) the "Log-in Help" points. The default file can be replaced with a custom HTML file containing information pertinent to your institution. In addition, versions of the custom file, translated into the twelve languages supported by LON-CAPA can be uploaded, and the link will automatically point to the appropriate (localized) file, depending on the viewer's language preference (as reported by the client web browser).

Where the "Contact Helpdesk" web form is in use it can be configured to include a CAPTCHA mechanism to discourage robotic form completion. There are two types of CAPTCHA to choose from – the "original" CAPTCHA which uses a self-contained perl module included with the LONCAPA prerequisites, or ReCAPTCHA, which uses an external web service – https://google.com/recaptcha – and requires you to create an account and generate public and private keys which will be entered in the domain configuration form. If you have more than one server in your domain, you should request "global" keys, as the same keys will be used by the Contact Helpdesk ReCAPTCHA on all servers in your domain.

The head portion of the log-in page may contain custom mark up (e.g., a script block containing javascript for page analytics) in a file which will be uploaded and published public. Different custom markup may be uploaded for each server in a domain, and a comma separated list of IP addresses may be specified for which the custom markup will not not be included in the page, when the request for the log-in page originates from one of those addresses. A use case for the exempt IP addresses is where robotic requests for the log-in page and made from a monitoring machine, used to detect when a LON-CAPA server is not working correctly.

## 2.2 Setting E-mail Addresses for Administrators and Support

LON-CAPA will send automatic e-mail to administrators/support staff under certain circumstances. The contact information data table can be used to provide e-mail addresses for receipt of these e-mails and to configure which types of e-mail should be sent to each address.

The types of e-mail are:

- Error reports - whenever a server encounters a 500 error (Internal Server Error), Apache will handle that event by displaying an error report form which the affected user can complete and submit. The submission, which contains session information besides any information provided by the user, will be sent as an e-mail.
- Package update alerts - the CHECKRPMS script run every other day will generate e-mail if it detects that package updates are needed.
- Helpdesk requests - clicking the Help link displayed at the right side of the inline navigation bar at the top of a LON-CAPA page (unless the Remote Control is active) will display a Help Menu which includes an "Ask helpdesk" link. The "Ask helpdesk" link provides access to a web form which a user will complete and submit to request LON-CAPA support. The submission, which contains information about the user's browser, besides information provided by the user, will be sent as an e-mail.
- Module integrity alerts and new LON-CAPA releases - a nightly process compares checksums for LON-CAPA modules installed for your current release with the expected values for that release retrieved from one of the Academic Consortium servers (if your server is part of the LON-CAPA network). An alert is sent if checksums or versions don't match, or if a new LON-CAPA version has been announced.
- Alerts for users sharing the same student/employee ID - a script is run every other night to check for use of the same student/employee ID by multiple users in your domain. In LON-CAPA, a username must be unique, and it is also recommended that student/employee IDs are unique.

Definition of the default Admin e-mail address and the default Support e-mail address saved from the "Contact Information" screen supercede any definitions made when ./UPDATE is run to update to a new version of LON-CAPA. Addresses entered the first time ./UPDATE was run on the primary library server for the domain (i.e., when LON-CAPA was first installed) will continue to apply until the first "Save" of the Contact Information settings has occurred in the domain.

Two additional settings allow you to indicate whether error reports should be sent to the LON-CAPA developers at Michigan State University (as well as to the recipients selected in your domain), and the same for an e-mail reporting a completed upgrade to a new LON-CAPA version. The default is to send both types of e-mail to the developers.

## 2.3 Default Color Schemes

Default color schemes can be set for the domain for four types of user context:

- Student
- Course Coordinator
- Author/Co-author/Assistant Author
- Domain roles (Domain Coordinator)

In each case the following can be set or modified:

- Header image displayed in place of the inline navigation controls when the page is viewed with the Remote Control active.
- Color used for text in the LON-CAPA interface
- Background colors used for the page itself, and the inline navigation (if shown) and the page header below it, and a border to the header (not used).
- Default colors for links in the page, depending on status: either active, visited or default (if neither apply).

Individual users can override any default settings you establish for the domain via the "Change Color Scheme" link in their individual Preferences screen. In the absence of individual preferences, any domain defaults you set will be used whenever users from your domain are using LON-CAPA, regardless of which domain owns the server hosting the session.

## 2.4 Setting Domain Defaults: Authentication, Language, Time Zone, Portal and User Types

Prior to LON-CAPA 2.7, default language and authentication type/argument were defined in the domain's entry in the domain.tab file. Those settings will continue to be used by servers in your domain until you have displayed and saved the Default authentication, language, timezone data table. Once that has been done, whenever values need to be determined for these settings in the domain they will be retrieved from the configuration.db file on the primary library server in your domain, which is where information saved from the "Domain Configuration" data tables is stored. Any information in the domain.tab file will no longer be consulted, except by servers running pre-2.7 versions of LON-CAPA.

Default domain configurations can be assigned for:

- default language used by users in your domain, unless overridden by a user preference
- default authentication type for new users in the domain. You will need to set the default authentication if you intend to allow a user to create a LON-CAPA account if the user successfully authenticated via a central service at your institution (e.g., Kerberos), but is without a LON-CAPA account. The default authentication is also the default offered when Course Coordinators or Authors create new accounts, assuming user creation is permitted in these contexts.
- default timezone - this will be the timezone used when showing any times in your domain, unless overridden at a course level, by a course-wide timezone. The timezones available are mostly in the form Continent/City, although for the USA there are some in the form America/State/City as well as EST5EDT, CST6CDT, MST7MDT, PST8PDT and HST (for Eastern, Central, Mountain, Pacific and Hawaii Timezones, which adjust for daylight savings as appropriate). If no default timezone is set times will be displayed according to the timezone of the server hosting the user's LON-CAPA session.

- portal/default URL - starting with LON-CAPA 2.10, a default URL can be specified. This URL will be included in e-mail sent to confirm self-enrollment etc. and might be for a load-balancer LON-CAPA server, or in the case of a multi-domain server, for a specific alias used for the domain.

Domain configurations can also be set for institutional user types via the same screen.

Prior to LON-CAPA 2.11, institutional user types were defined in the &inst_usertypes subroutine in localenroll.pm, which would be customized for consistency with types defined in institutional data feeds. Setting of user types via the Domain Configuration web GUI supersedes use of localenroll::inst_usertypes(). Items that can be set are:

- Internal ID (e.g., faculty)
- Name Displayed (e.g., Faculty/Academic Staff)
- Order (Listing order, 1 through N, when the type is to be selected from a list).
- Whether status type can also be assigned to a non-institutional user with an e-mail address as username

## 2.5 Setting Domain Defaults: Blogs, Portfolios, Personal Information Pages, WebDAV, and Quotas

By default, each user in your domain can create blogs, a personal information page, and store files in an individual portfolio space. Students can submit items from their portfolio to meet the requirements of assignments in their courses.

You can choose to disable personal information pages, blogs and/or portfolios for different groups of users defined for your domain (e.g., Faculty, Adjunct, Staff, Student). If the "Modify User" utility in User Management is used to explicitly set availability of these tools for a particular user, that will override the corresponding settings determined by the user's affiliation.

If you choose to enable portfolios, default quotas (in MB) can similarly be set to vary by institutional affiliation. If a user is affiliated with more than one group, whichever default quota is largest for the different groups is the one which applies. Institutional types are defined in the "Institutional user types" section on the "Default authentication, language, timezone, portal, types" screen. If no types have been defined, then a single default quota will apply for all users from the domain.

Default portfolio quotas which can be set for users in your domain will be overridden by any quota you set for an individual user via: the "Modify User" utility.

Additional options for authoring spaces can be set for the various user types: (a) whether webDAV is active, and (b) the default quota for Authoring Space. These only come into effect for a particular user, when an author and/or one or more co-author roles have been assigned to a user to provide access to one or more Authoring Spaces.

WebDAV allows a user to connect to an Authoring Space in the same way as removable media, and to use their own laptop/desktop computer's file management tools and applications to move, edit and delete files. See: "WebDAV access to Authoring Space" section in the Authoring manual for more information.

Note: webDAV usage requires Apache with SSL to be running on the library server. The user will be prompted to enter his/her username (this will be the LON-CAPA username or username,domain if the access is for a co-author with a domain different to that of the author), and the user's LON-CAPA password.

If you use Single Sign On to authenticate LON-CAPA users in your domain, then to support webDAV you also need to support authentication by LON-CAPA for your users. This can achieved if the authentication type stored internally for each SSO user is set to either (a) Kerberos 5 (with a parameter – the appropriate Kerberos realm set), or (b) Local Authentication, with /home/httpd/lib/perl/localauth.pm customized to verify username and password (e.g., via LDAP). If a user can log-in to LON-CAPA via the URL /adm/login (thereby by-passing SSO), then the same user will also be able to authenticate using a WebDAV client (assuming other requirements are met, i.e., SSL, WebDAV access enabled, active author or co-author role).

The requirement to obsolete published files before moving or deleting them, which applies to the regular web browser-based management of Authoring Space, also applies in the webDAV environment. Moving and deleting directories in the webDAV environment is prohibited if the directory, or any (nested) subdirectory contains a non-obsolete published resource.

Given the ability to easily delete unpublished content in webDAV (without the ability to reverse the deletion), it is important that authors assigned webDAV access are especially careful when working in the webDAV environment.

## 2.6 Identity Management: Searching for Users in an Institutional Directory

LON-CAPA provides a mechanism to search for users by complete username, last name, or last name,first name (or fragments of each). Accounts within the LON-CAPA domain itself can be searched, or alternatively, an institutional directory may be searched, if you are able to implement a conduit to directory information to support such searches. This will be done by customizing the &get_userinfo() routine in localenroll.pm. This routine has a dual role in LON-CAPA. Not only will it support searches where the user is not excatly known, but it can also be used to retrieve institutional records for a specific username or student/employee ID. This latter mode of use is appropriate when adding a new user to LON-CAPA.

Settings for directory searches are:

- Set directory searches as available or unavailable in the domain
- Set whether users from other LON-CAPA domains can use the institutional directory search to search for users. Note: this only applies to institutional directory data, users with privileges to add users to a course will always be able to search the LON-CAPA user database for other domains.
- For searches by users within your domain, you can limit the users who can use directory search based on institutional status

- Set which types of search method - username, last name or last name,first name are allowed for institutional searches
- Set which types of search latitude - exact match, begins with or contains - are allowed

In the case of a "contains" type search at least three characters must be entered by the user as the search term. Searches are case insensitive.

## 2.7 Identity Management: Creating New Users

Identity management in a LON-CAPA domain is dependent on settings made for user creation and user modification. Of particular concern is the potential for assignment of usernames in a format used by your institution when the username does not yet exist. In such a case, authentication is likely to be set to be "internal", and should a real user be created in the future, and be enrolled in a course by auto-enrollment, the user would either be unable to authenticate (using LON-CAPA log-in page), or would be authenticated by SSO, and have access to the original user's roles and associated information.

It is important therefore to establish format rules for new usernames so the only users created with institutional-type usernames are the real users themselves with the appropriate authentication type (Kerberos or localauth). Even without format rules, the Domain Coordinator can set who can create new users, and the authentication types that may be set in different context.

The domain-wide options available for user creation are:

- Activate/deactivate operation of format rule(s) for usernames
- Activate/deactivate operation of format rule(s) for student/employee IDs
- Control which types of username (official or non-official) may be used when creating new users in course or author context
- Control which types of authentication may be used when assigning authentication to new users in author, course or domain context

The format rules themselves are defined by customizing the following routines in localenroll.pm:

- usernames: &username_rules() and &username_check()
- IDs: &id_rules() and &id_check()

When enforced the user name and ID rules require that if a username and/or ID which matches the format for an active rule is to be used in LON-CAPA, they must exist in the institutional directory. If they exist, the corresponding user information (first name, middle name, last name, e-mail address) will be used when creating the new user account. If they do not exist, account creation will not occur.

Domain-wide settings are available to set whether users may create their own accounts, and if so, which types of users may do so, and what types of user information a user self-creating an account will provide.

In the case of an institutional login, or single sign on (SSO), a user must first authenticate with an insititutional username and password. If user information is available from an institutional data source via a query using the username (mediated via a customized routine in localenroll.pm), then that is used to populate appropriate data fields in the user's new LON-CAPA account. A domain configuration is available to specify which fields the user may self-report, if the corresponding institutional data are unavailable.

Which types of institutional log-in may self-create accounts can be restricted, institutional status (e.g., Faculty, Staff etc.). The institutional status types are set in the "Default authentication, language, timezone, portal, types" item in the domain configuration (this is a change from 2.10 and earlier, which used custom routines in localenroll.pm for that).

For Shibboleth SSO users, mapping of Shibboleth environment variable names to user data fields can be set, so that the appropriate user information is stored at account creation time.

Self-creation of user accounts may also be enabled for non-institutional login. In this case the user will provide an e-mail address as a username, and will also set a password. The user must have access to e-mail sent to that address, as completion of the account creation process requires submission of a link (containing a token), sent to the e-mail address.

In order to discourage creation of multiple accounts by a single user when self-creation is available in a domain for both insitutional log-in and e-mail address as username, a domain may want to consider implementing format rules which prohibit self-created accounts from using certain types of e-mail address as the username.

If a user attempts to self-create an account employing a username with an e-mail address in a format which matches a defined rule, the action does not proceed, and the user is directed to create an account with the corresponding institutional log-in. In this case, account creation can only occur once the user has authenticated using that log-in.

Self-created accounts with an e-mail address as username can be set to be queued for approval or created automatically. Institutional status types can be set to be self-reported for e-mail type usernames – set in the "Default authentication/language/timezone/portal/types" area – and processing (queued or automated) can be set, based on status.

User information (in addition to e-mail address and password) can be set to be required, optional or not requested.

A Captcha mechanism can be used to validate that the user requesting a self-created account is a person, not a script. There are two types of CAPTCHA to choose from – the "original" CAPTCHA, which uses a self-contained perl module included with the LONCAPA prerequisites, or ReCAPTCHA, or ReCAPTCHA, which uses an external web service – https://google.com/recaptcha – and requires you to create an account and generate public and private keys which will be entered in the LON-CAPA domain configuration form. If you have more than one server in your domain, you should request "global" keys on the google.com/recaptcha site, as the same keys will be used by the Account creation form's ReCAPTCHA on all servers in your domain.

## 2.8 Identity Management: Modifying Existing Users

Configuring settings which apply to modification of existing user information (names, e-mail address, student/employee ID) form a part of LON-CAPA identity management. Authors and Course Coordinators both have access to "Manage Users" which permits them to assign roles to users appropriate to the context. In addition to the ability to assign roles, the ability to modify existing user information may be conferred in this context depending on the target user's role(s). The types of user information which are modifiable in the different contexts is also configurable.

## 2.9 Identity Management: Automated Updates of User Information

An auto-update, run as a regular process, can update user information stored in LON-CAPA for all users in a domain, for whom institutional directory information is available. Which user records are updated can be controlled by institutional status (e.g., Faculty, Staff, Student etc.). If a user is affiliated with more than one group, then the attributes which can be updated will be the cumulative set for the different groups to which the user belongs.

If users are not affiliated with any institutional group, they can be accommodated within the default "Other users" group which is provided automatically. If no status types are defined for your domain, this default group is entitled "All users".

Settings for auto-update are:

- Set auto-update as active or inactive in the domain.
- Set whether user information changes should propagate to data stored in classlist database files for the separate courses in which the user has an active student role.
- Set whether users with a particular status (e.g., Faculty, Staff, Student etc.) should have access to a user preference which permits them to lock their existing user information, and disable automatic updates of their own information, should it change in the institutional directory. Note: this option is only shown if institutional groups have been defined.
- Set which of the following attributes: first name, middle name, last name, generation, e-mail address, student/employee ID should be updated within LON-CAPA if a different version to the one currently stored is retrieved from the institutional directory.

In order for Autoupdate to work, the &allusers_info() routine in localenroll.pm needs to be customized and a conduit established to institutional data. In addition, if you wish to differentiate between institutional user types in your LON-CAPA domain, you should define those in the "Institutional user types" section of the "Default authentication, language, timezone, portal, types" domain configuration screen. The types you set should be consistent with the types in use at your institution. These types are then used to populate the "User population" column in each of the "Updatable user information" row(s) in the Auto-update data table in "Domain Configuration".

Warnings will be written to the Auto-update log file found in /home/httpd/perl/logs if a possible username change is detected. Although the username is the unique identifier in LON-CAPA, the student/employee ID operates as an additional, mostly unique identifier. At present LON-CAPA does not support username changes. For users who switch username (assuming institutional authentication will no longer authenticate the user's old username) the recommendation is to convert the authentication type in LON-CAPA for the user to "internal", set an initial password, make sure that permanent e-mail is set for the user, then e-mail the user and ask them to use the "Forgot password?" link on the log-in page to change the password to something secure.

## 2.10 Automated Course Creation

An entry is included in the loncapa file in /etc/cron.d which will automatically run the Autocreate.pl script (by default at 2.30 am local time) on LON-CAPA library servers. The behavior of the script is controlled by the Auto-course creation settings set for a domain.

The script can create pending courses from two locations:

- Pending requests (XML format) from any pre-LON-CAPA version 2.9 custom web forms created at your institution

  This may not apply for your domain. Course descriptions in XML form (located in /home/httpd/perl/tmp/addcourse/$dom/auto/pending), which are typically when an institution has created its own online course request form which faculty use to request courses. Prior to LON-CAPA version 2.9, this was the only way to automate course creation.

  The format of the XML used in the course description files is the same as used for batch files of course requests which can be uploaded by a Domain Coordinator. (See: **??** Batch Creation).

- Queued course requests for official courses, pending validation

  If you have established a conduit to an institutional data source which permits validation of instructor of record status, then you can configure the LON-CAPA course course request form to require instructor validation for official courses. If an instructor requests creation of a course for which he/she is not currently listed as an instructor of record, the request will be queued. The Autocreate.pl script can check each queued request to see if the status of the requestor has changed.

  If the requestor is now validated as an instructor of record, the course request will be processed.

## 2.11 Course/Community Requests

A domain configuration can be used to determine which users in the domain may request creation of: (a) official courses, (b) unofficial courses, (c) communities, or (d) textbook courses within the domain.

The default is for no course or community requests by any domain users.

Communities are similar to courses except the Coordinator may only browse areas of the shared LON-CAPA repository for which he/she has an author or co-author role. Textbook courses use a simplified (1 page) request form, in which the options include cloning of available pre-fabricated textbook courses or template courses, for which the requester has cloning rights.

Course/Community requests may be set to be processed automatically, queued for approval by a Domain Coordinator, or (for official courses, for example) validated against institutional instructor of record data. The "With validation" option is only displayed if the crsreq_checks() subroutine in localenroll.pm has been customized to indicate that a particular course type can be validated if the owner belongs to a specific institutional group (e.g., Faculty).

The crsreq_checks() routine is closely tied to validate_crsreq(). "With validation" should not be a possible choice in the domain configuration menu for a particular course type and institutional affiliation, unless corresponding validation code has been implemented in validate_crsreq().

LON-CAPA can be configured to send messages to Domain Coordinators to notify them when a course has been requested and queued pending approval. Recipients of such messages are selected from users with an active Domain Coordinator role for the domain. These same users will also receive messages when a course request has been approved by a Domain Coordinator.

LON-CAPA messages are sent to course requestors when a queued course is approved, or an official course which had been queued pending validation of instructor status, is created when validation occurs.

The course request settings for each course type may be different for the various affiliations defined for the institution (e.g., Faculty, Staff etc.). In addition, settings may be defined for users with "advanced" roles in LON-CAPA (i.e., users with author, co-author, coordinator, instructor or administrator roles) which will override the course request settings based on affiliation.

The "default" settings which apply to a particular user based on domain configuration, may be overridden for that specific user, by a Domain Coordinator in the user's domain. The "Add/Modify a User" page for a user includes a section where custom settings can be selected for requests for creation of Courses/Communities in the domain.

An individual user may still be able to request courses or communities, even if settings preclude that in the user's home domain. Domain Coordinators displaying the "Add/Modify a User" page for an existing user from a different domain will see a section where custom settings can be selected to allow requests for creation of Courses/Communities in the domain for which Domain Coordination is being carried out.

Details of pre-fabricated courses: "textbooks" and "templates" for cloning when creating a new "textbook" course can be uploaded (subject, title, image, author, publisher, and LON-CAPA course domain/ID).

Validation of unofficial and textbook courses, and communities can be via a validating server/script Form fields to send to the validator can be specified by customizing the &crsreq_updates() subroutine in localenroll.pm.

## 2.12 Course Catalogs

The availability of a course/community catalog, as well as the type of catalog provided are both domain configurations. A choice of catalog can be set for two contexts: (a) public-facing web page (i.e., unauthenticated users), and (b) web page for authenticated users.

For each context the options are:

- Standard catalog (the only type available pre-LON-CAPA 2.11), which contains both self-cataloging courses, and manually cataloged courses.
- Domain-only catalog, which is the same as the standard catalog, except no domain selector is available to switch to display course listings in other domains.
- Code-search form, which provides a web form to search for a course by the unique six character code, which can be set to be assigned at course creation.
- No catalog

In the standard and domain-only catalogs, self-cataloging uses the institutional course code assigned to the course when it is first created, or when the course is modified by a Domain Coordinator via "Modify a course". If a course has no institutional code it will not appear in the category: Official courses (with institutional codes).

A hierarchy of categories and sub-categories can be defined which are independent of institutional course codes. These categories might be used to catalog courses in the domain to which the "official courses" designation does not apply, or they might be used to provide alternative ways of cataloging official courses.

Besides definition of the hierarchy of categories and sub-categories, the "cataloging of courses" screen provides two options to be set for the domain by a Domain Coordinator, which control:

- Who may hide a course from the course/community catalog, if the course would ordinarily appear by virtue of having an institutional course code, or having been assigned to a custom category/sub-category.
- Who may assign a custom category/subcategory to a course

In both cases, the choice is between a Domain Coordinator and a Course Coordinator. For the former, hiding of courses and assignment of categories will be via "Modify Course", while for the latter these operations will be via "Modify Parameters" ("Set Course Environment" option in sub-menu).

Definition of custom categories is by the Domain Coordinator. The "Cataloging of courses" interface allows custom categories and sub-categories to be defined and reordered. There is one category listed at the top level in the hierarchy which behaves differently to the others - the category: "Official courses (with institutional codes)". This is the category which is used for self-cataloged courses: the option is to either display or not display.

Although sub-categories can not be defined via this interface for this "system" category (unlike the other "custom" categories), customization of two routines in localenroll.pm -

&instcode_defaults() and &instcode_format() are used to automatically generate linked select boxes which will be displayed when the course/community catalog is shown to allow users to include limits to their searches for official courses. If these routines have not been customized, the catalog for official courses will display all courses with institutional codes, which have not been specifically hidden.

When &instcode_format() has been customized it will populate perl structures (hashes and arrays) which LON-CAPA will use to generate the Javascript code embedded in the course/community catalog page which is used in the functioning of the linked select boxes. The contents of the hashes and arrays are determined from the complete list of institutional course codes used in the domain. For example at MSU, the following linked select boxes are displayed for the official course/community catalog: Year, Semester, Department, Number.

The course/community catalog is useful in domains where Course Coordinators have opted to allow self-enrollment in their courses. In such cases, students can include a flag to only display courses allowing self enrollment when they display courses from the catalog. Course Coordinators will be advised, when enabling self-enrollment, if a course is currently unlisted in the course/community catalog (and therefore difficult for students to locate), and the action to take to rectify the reason why there is no listing, which could be because:

- the course is hidden
- the course has no institutional code
- the course has not been assigned to any custom categories
- the course has an institutional code, but the course/community catalog has been set to not display self-cataloged courses.

## 2.13 Automated Enrollment in Official Courses

If your institution can provide access to roster information for courses using LON-CAPA then your domain can offer automated enrollment once localenroll.pm has been customized on each of the library servers in your domain which serves as a home server for one or more courses. The required customization is the creation of connections to rosters for institutional course sections providing enrollment to each LON-CAPA course. These connections can involve queries of database tables, in real time, or can involve retrieval from a data source which is only updated periodically.

There are three configuration options:

- Set auto-enrollment as active or inactive in the domain.
- Set the username:domain used in the notification messages sent when changes in enrollment occur as a result of auto-enrollment updates. By setting these to a specific user (you might create one for this purpose), you can view all auto-enrollment change messages for the entire domain by viewing the contents of the sent messages folder for that user.

- Automatically assign co-ownership If this is set to yes, then whenever a Course Coordinator role is assigned in a course with an institutional code, a check is made to see if the user to whom the role is being assigned is officially listed as instructional personnel. If so, and the user is not the course owner, then the user will be identified as a co-owner. Co-owners are listed in the course/community catalog, and also in the pop-up window displayed when picking a course (e.g., for cloning, ). For the validation to work the validate_instcode() routine in localenroll.pm must have been customized to include the username supplied as the third argument in the query made to the institutional data source which ties instructors to institutional codes.

Auto-enrollment settings for each course consist of items set by a Domain Coordinator within the "Modify Course" area, and those items set in a course context from the "Automated Enrollment Manager" link in the "Manage Users" menu. This link is only displayed if auto-enrollment has been set to be active in the domain.

The items which must be set by a Domain Coordinator include:

- institutional code (used in mapping institutional rosters to LON-CAPA courses)
- default authentication
- course owner

Your institution may have policies in place to control who may have access to student information contained within course rosters. In such cases, assignment of an appropriate course owner in LON-CAPA may facilitate access to institutional rosters. When a course is first created the initial Course Coordinator chosen will be identified as the course owner. If, instead a course is created using an uploaded XML course description file, the XML file will include tags to define the username and domain of the course owner.

Settings which control auto-enrollment which are modifiable by a Course Coordinator are described in the **??** help page.

The auto-enrollment process will update user information (name, e-mail address, studentID etc.) for any student who is being added to the course, but will not change it in other case (i.e., drops, section switches) should there be a difference between the values in the institutional roster and the values in the LON-CAPA classlist. To change user information for students already in the course, the Auto-update process must be run (see below).

Warnings will be written to the Auto-enrollment log file found in /home/httpd/perl/logs if a possible username change is detected. Although the username is the unique identifier in LON-CAPA, the studentID operates as an additional, mostly unique identifier. The same studentID may not be assigned to more than one user - if an existing studentID is assigned to adifferent user, the change will not occur, unless forced (there's a checkbox for that). A blank studentID can be assigned, and in this case multiple users can share the same studentID. StudentIDs are used for LON-CAPA grading of bubblesheets, and are also used to detect changes in username by the auto-enrollment and auto-update processes.

At present LON-CAPA does not support username changes, although this functionality will be supported in the future. In the meantime, what you must do for users who switch username mid-semester (assuming institutional authentication will no longer authenticate

the user's old username) is to convert the authentication type in LON-CAPA for the user to "internal", set an initial password, make sure that permanent e-mail is set for the user, then e-mail the user the initial password, and ask them to use the "Forgot password?" link on the log-in page to change the password to something secure.

## 2.14 Course/Community Defaults

Starting with LON-CAPA 2.10, a Domain Coordinator can configure default settings for courses in the domain. Defaults are of two types:

- defaults that can be overridden in an individual course by a Course Coordinator using Settings -> Course Settings -> Display ("Display of resources" checked).

  Currently, there are two domain defaults of this type:

  - Student formula entry uses inline preview, not DragMath pop-up

    For problems that require a formula input via a textbox, the default is to show the math expression in a previewer below the textbox, as the student types. The alternative is to use the older (Java-based) DragMath pop-up which allows the student to compose the expression in a WYSIWYG-style editor in a pop-up window and then save it to the textbox in the main window using a "Save and Close" button.
  - Molecule editor uses JSME (HTML5) in place of JME (Java)

    For problems that require the student to "draw" an organic structure using a molecule editor, the default is to use the HTML5 version (JSME). The alternative is to use the older (Java-based) JME editor.

- defaults that can be overridden in an individual course by a Domain Coordinator, via the "View or modify a course or community" interface.

  Currently, there are four domain defaults of this type:

  - responder count - number of responses needed before submissions made to anonymous surveys (with no identifying information) are viewable by Course personnel.
  - default course quotas - domain default for quotas (MB) for official, unofficial, community, and textbook course types for content uploaded directly to the course.
  - default course credits - whether credits may be specified for courses (Yes/No), if yes: default values for number of credits for official, unofficial and textbook courses.
  - submit button behavior post-submission

    The default is for the "Submit Answer" button in the current page to be disabled for 60s following answer submission. After processing a student's answer a new page is displayed. The setting to disable a button, (with or without a timeout), applies to the button in the old page in the time period before the new page is loaded by the browser.

## 2.15 Course/Community Self-enrollment

Ordinarily, course personnel (e.g., Coordinators) have the right to configure self-enrollment settings in a course to determine which types of LON-CAPA user (if any) may request enrollment in their courses (and when), and how such requests will be handled.

However, the domain can be configured such that control of self-enrollment in courses is a domain coordinator action. Control over self-enrollment is very granular - each of the seven settings can be set, by default, to be under the control of domain coordinators, or under the control of course coordinators. In addition domain-wide default settings can be overridden for each of the seven, on a course by course basis, by using the "View or modify a course or community" utility from the Domain Coordinators' main menu, to select a course, in which to set different choices.

The seven self-enrollment settings are:

- Users allowed to self-enroll
- Registration status (official courses)
- Dates self-enrollment available
- Access dates for self-enrolling users
- Self-enrolling users' section
- Processing of requests
- Enrollment limit

The options for processing of self-enrollment requests are to: (a) process automatically, (b) queue pending approval, and (c) queue pending validation.

In the case of self-enrollment queued pending validation, the validation itself can be carried out via an external validation server/script. Validation is configured at the domain level, by providing: (a) the web address of the validation server (and/or script), the form fields to send to the validator – one or more of: username, domain, six-character code, course Id, coursetype (i.e., official, unofficial, community, or textbook course), and course descripton (i.e., title). The text used for the button displayed when a student has submitted a self-enrollment request (for validation), as well as accompanying text (HTML format) can also be set in the domain configuration.

## 2.16 Authoring Space Requests

A domain configuration can be used to determine which users in the domain may request creation of an Authoring Space.

The default is for no Authoring Space requests by any domain users.

Authoring requests may be set to be processed automatically or queued for approval by a Domain Coordinator.

LON-CAPA can be configured to send messages to Domain Coordinators to notify them when an Authoring Space has been requested and queued pending approval. Recipients

of such messages are selected from users with an active Domain Coordinator role for the domain. These same users will also receive messages when an Authoring Space request has been approved. A LON-CAPA message is also sent to the requestor when a queued request is approved.

Authoring Space request settings may be different for the various affiliations defined for the institution (e.g., Faculty, Staff etc.). If users are not affiliated with any institutional group, they can be accommodated within the default "Other users" group which is provided automatically. If no status types are defined for your domain, this default group is entitled "All users".

In addition, settings may be defined for users with "advanced" roles in LON-CAPA (i.e., users with co-author, coordinator, instructor or administrator roles) which will override the course request settings based on affiliation.

The "default" settings which apply to a particular user based on domain configuration, may be overridden for that specific user, by a Domain Coordinator. The "Add/Modify a User" page for a user includes a section where custom settings can be selected for requests for an Authoring Space.

## 2.17 Bubblesheet Data Formats

Where bubblesheet exams are used in a course, the format of data in the file generated from the processing of bubbled-in bubblesheets may vary between institutions. The format definitions available when performing bubblesheet grading in LON-CAPA were originally listed in the scantronformat.tab file, stored in /home/httpd/lonTabs, which might have been modified locally on each server.

Starting with LON-CAPA 2.7, bubblesheet format information is read from either a custom.tab file, or a default.tab file both of which belong to the special domain configuration user ($dom-domainconfig, where $dom is the name of the domain) and which are automatically published into resource space.

For LON-CAPA installations older than 2.7, when the primary library server for the domain has been updated, a Domain Coordinator should display the "Bubblesheet format file" configuration page via "Domain Configuration". The first time this page is displayed, a default.tab (a copy of the standard LON-CAPA scantronformat.tab file), and a custom.tab (if the scantronform.tab file currently on the server differs from the standard file) will be copied and published. Thereafter any changes to bubblesheet format files to be used for grading bubblesheet exams in courses from the domain will be made via the Domain Configuration menu. Any scantronform.tab files in /home/httpd/lonTab directories on servers in the domain will no longer be used.

The settings available via "Bubblesheet format file" support upload of a new custom file, or deletion of an existing custom file (in which case grading will default to use of the default.tab file). An uploaded bubblesheet format file contains one or more lines of colon-separated values for the parameters in the following order:

name:description:CODEtype:CODEstart:CODElength:IDstart:IDlength:Qstart:Qlength:Qoff:Qon:PaperID:PaperIDlength:
FirstName:FirstNamelength:LastName:LastNamelength

1. *name* is the internal identifier used within LON-CAPA

2. *description* is the text displayed for each option in the "Format of data file" dropdown in the Bubblesheet grading screen. The user will choose the appropriate format for the bubblesheet file currently being used for bubblesheet grading.

3. *CODEtype* can be either 'none' 'letter' 'number'

4. *CODEstart*: (only matters if a CODE exists) column in the line where the CODE starts

5. *CODElength*: length of the CODE

6. *IDstart*: column where the student ID starts

7. *IDlength*: length of the student ID

8. *Qstart*: column where the information from the bubbled 'questions' start

9. *Qlength* is the number of characters in the raw data used to record the student's bubbled answer for each row on the bubblesheet.

   If the raw data indicate bubble position by a letter or a number (e.g., A = first bubble or 1 = first bubble etc.) then the number of characters would be 1 (assuming 10 bubbles or less for the number case). If however, bubble position is indicated by position of the Qon (bubbled) character, then each bubble row might have 10 characters (one for each of the 10 possible positions with Qon at the position bubbled and Qoff elsewhere, (e.g. if Qon = 1, Qoff = . the portion of a student's record for a single bubble row might be: ...1..... indicating position 4 was the one bubbled.

10. *Qoff* is the character used to indicate an unfilled bubble, this is commonly a single blank space.

11. *Qon* is the character used in the raw data generated by the bubblesheet scanner to indicate a filled-in bubble.

    *Qon* can be either:

    - the symbol that says a bubble has been selected, or
    - 'letter' (for when the selected letter appears), or
    - 'number' for when a number indicating the selected letter appears

12. *PaperID*: if the scanning process generates a unique number for each sheet scanned the column that this ID number starts in

13. *PaperIDlength*: number of columns that comprise the unique ID number for the sheet of paper

14. *FirstName*: column that the first name starts in

15. *FirstNamelength*: number of columns that the first name spans

16. *LastName*: column that the last name starts in
17. *LastNamelength*: number of columns that the last name spans

As an example, below are four different format lines: the first two were used at MSU prior to 2006; the last two have been used since then.

- msunocode:MSU without any CODE:none:0:0:57:9:77:10: :1:5:5:51:1:41:10
- msucode:MSU with CODE in separate location:letter:69:6:57:9:77:10: :1:5:5:51:1:41:10
- msucodelet:MSU with CODE in separate location (letter format):-1:69:6:57:9:77:1: :letter:5:5:51:1:41:10
- msucodenum:MSU with CODE in separate location (number format):-1:69:6:57:9:77:1: :number:5:5:51:1:41:10

## 2.18 Access to Server Status Pages

Access to pages which provide server status information for LON-CAPA servers in a domain is controlled by a Domain Coordinator. When a user who has a Domain Coordinator role is logged into LON-CAPA, the user automatically has access to server status pages. Access can also be granted to other users by either (a) specifying username:domain, or (b) specifying IP addresses for the clients from which access will be made. In the case of IP-based access there is no need for the user to be logged into LON-CAPA or even have a LON-CAPA user account.

- There are eleven server status pages:
  - User Status Summary - information about User Sessions which have been hosted on the server since the last nightly clean-up of lonIDs for stale sessions, and where the user has not logged out. Sessions are classified into: Active, Moderately Active and Inactive.
  - Detailed Report - information saved by the nightly run of loncron which checks connections to other servers in the cluster, and includes excerpts from various logs, as well as machine information.
  - Apache Status Page - information from the Apache web server about its current status
  - LON-CAPA Module Versions - a list of currently installed LON-CAPA perl modules, including version information.
  - LON-CAPA Module Checking - the results of comparison of version number and checksum for installed LON-CAPA modules, with expected values for the LON-CAPA version (from a data file for the specific release number, available from any of the LON-CAPA Academic Consortium servers).
  - Domain status - information about the status of LON-CAPA daemons for servers in the domain.

  - Show user environment - Information about the current user's session environment.
  - Text Display of Domain Configuration - Information about the domain's configuration (essentially a dump of the domain's configuration.db GDBM file).
  - Six-character Course Codes - a listing of course information and codes, for courses for which a six character code has been assigned.
  - Course/Community Disk Usage - Quota (MB) allocated for content uploaded directly to each course, usage in MB and percent (with choice of courses to display filtered by standard course picker).

- There are five pages which can be used to perform server actions:
  - Generate Detailed Report - run the loncron command which checks connection to other servers and creates a new version of the detailed server status report.
  - Offline: replace Log-in page - replace the standard LON-CAPA index.html page at the Document Root with a temporary page announcing unavailability of LON-CAPA service on that particular server. It is strongly recommended that access to the corresponding "Online - restore Log-in" page is set to allow access from your IP address so that you can visit that page to re-enable the standard index.html page (e.g., when maintenance is complete) if you use the Offline page to replace it.
  - Online: restore Log-in page - replace the temporary index.html page with the standard index.html (which redirects to /adm/roles – which will display the log-in page unless the requestor's browser has an unexpired LON-CAPA session cookie).
  - Toggle debug messages - set which users will have access to a link on the user preferences page to enable/disable display of debug output on screen when rendering a LON-CAPA resource. Note: Domain Coordinators automatically receive the toggle debug link, regardless of settings on the server status page.
  - Cause server to ping another server - cause the server to attempt to initiate a connection to another LON-CAPA server in the cluster.

- There are two pages which can be used to display Metadata keyword information for searches in the domain.
  - Display Metadata Keywords
  - Harvest Metadata Searches

One final setting is for "Toggle debug messages". This controls whether a user's "Set my user preferences" page will include a "Toggle Debug Messages" link which can be used to set display of debugging messages in LON-CAPA either on or off.

## 2.19 User Session Hosting/Offloading

Default domain configurations can be assigned for:

- Offloading sessions when busy, or prior to maintenance

  Starting with LON-CAPA 2.11, as Domain Coordinator you can use the web GUI to configure where your servers will offload user sessions (for new log-ins) when they are busy, if your server is a member of a cluster of servers.

  If you have not yet configured offloading then the least busy of the servers listed in /home/httpd/lonTabs/spare.tab on each server will be used, if your server is found to be at over 100% user load or 100% server load.

  The default spare.tab file included with a LON-CAPA release includes four access servers in the msu domain, and one access server in the csm domain. If your LON-CAPA domain is part of the LON-CAPA network then one of the five servers will host offloaded sessions from busy servers in your domain. Once offloading has been configured using the GUI, the spare.tab files on your servers will no longer be consulted, since offload settings for your domain will now be stored centrally in the configuration.db file for your domain. Changes made via the web GUI are propagated immediately to all servers in your domain.

  The load limits which are used to compute percentage load are currently set on each server when you run UPDATE to install LON-CAPA (or update to a new release). The defaults are: Server Load: 2.00; User Load: 0.

  These values can also be set by editing the /etc/httpd/conf/loncapa.conf file (CentOS, Scientific Linux, RHEL, Fedora) or the /etc/apache2/loncapa.conf file (SuSE, SLES, Ubuntu), and setting values for lonLoadLim and/or lonUserLoadLim. (Web server reload required after making changes to loncapa.conf).

  The percent server load is computed from: 100 * loadavg/lonLoadLim, and percent user load is computed from 100 * activeusers/lonUserLoadLim.

  If you do not wish to restrict the number of active user sessions on your server set the User Load (i.e., lonUserLoadLim) to 0.

  The loadavg comes from the one minute average for CPU utiization, i.e., the first number in the output from cat /proc/loadavg.

  The activeusers total is the number of user session files in /home/httpd/lonIDs which have modification times within the last 30 minutes.

  The "User session hosting/offloading" option in the domain configuration GUI allows you to set where user sessions will be offloaded, separately for each server in your domain. You can designate offload servers as either primary or default. Primary servers are considered first when seeking to offload a user session.

  Ordinarily, session offloading only occurs when a user first logs in. However, you can also choose to offload all active users from a particular server by checking the "Switch active users on next access" checkbox for that server. You might use this feature if you wanted to take an access server out of the pool of servers in your domain

for maintenance (all sessions are switched regardless of current load values, with the exception of Domain Coordinators).

Notes:

1. As this type of offload happens on page load, it requires the user to be actively interacting with the system (i.e., the web browser needs to request a page).
2. Any transactions resulting from the original page request (e.g., grading of a homework submission) will be completed on the server before the output containing the server switch function is returned to the client.

- Session Hosting

  Starting with LON-CAPA 2.10, as Domain Coordinator you can configure a domain to include constraints on where sessions for users from your domain may be hosted, and also which other domains may have their users hosted on your servers.

  If a LON-CAPA server is part of a cluster in which there is a only a single domain, or multiple domains but only a single library server, then options to control user session hosting are unavailable, as they do not make sense in this context.

  - Hosting of users from other domains.

    There are two types of setting: "Allow all, but exclude specific domains" or "Deny all, but include specific domains". In both cases the options are (a) for the setting to be in use, or (b) not be in use (the default). If in use, then checkboxes can be checked for any "internet domains" for which the constraint is to apply. Internet domains encompass all servers at a particular institution, and also any aliases used on a multiple domain server. For example, there is a single internet domain for educog.com. Constraints for that internet domain will apply to all *.educog.com servers, as well as all domains on the multi-domain educog server. On a multiple domain server, session hosting constraints are defined in a single domain - the default domain included in the loncapa.conf file (e.g., the "author" domain for "educog.com").
  - Hosting domain's own users elsewhere.

    Again the same two types of setting are available: (a) "Allow all, but exclude specific domains" or (b) "Deny all, but include specific domains". There is a third type of setting: "LON-CAPA version requirement", which, in common with the other two can be set to be: "in use", or "not in use" (the default). This setting can be used to require that sessions for users in your domain are not hosted on LON-CAPA versions which predate a particular selected version.

## 2.20 Load Balancing with Dedicated LON-CAPA Server

A "Dedicated Load Balancer(s)" option will be included in the list of modifiable domain settings if your LON-CAPA domain contains more than one server or VM.

If you chose to designate one of your servers/VMs as a load balancer, you can configure balancing behavior as follows:

- Default destinations

  This determines which of your servers will host user sessions (by default) after log-in via the load balancer server. You can separate destinations into “primary” and “default”. The least loaded of the primary servers will be selected, unless all primary servers are considered 100% loaded, in which case the least loaded of the default servers will be selected.

- User affiliation – overrides

  The choice of server/VM to host user sessions for particular types of user can be further configured, by choosing an action for particular user types. The chosen action will override the default behavior. The choice of user types includes the various affiliations defined for the institution (e.g., Faculty, Staff etc.). If users are not affiliated with any institutional group, they can be accommodated within the default ”Other users” type which is provided automatically. If no status types are defined for your domain, this default group is entitled ”All users”.

  In addition to type based on affiliation, there are additional types:

  - Advanced users from domain (i.e., users with author, coordinator, instructor or administrator roles)
  - Users from domain with author role

  If an institution runs a library server with multiple domains, the server selected to be the load balancer can be different from the particular domain for which balancing is being configured. Two other additional types are available where the load balancer’s domain and domain being configured match:

  - Users not from domain, but with a home server matching institution’s web address (e.g., yourcollege.edu)
  - Users not from domain or with a home server matching institution’s web address web address

  In all but the last case, the options available are:

  - Offloads to default destinations
  - Offloads to user’s home server
  - Offloads to a specific server in the user’s domain (select froma dropdown list)
  - No offload

  In the last case (i.e., users whose home domain is not on any of the servers/VMs you manage), the options are: (a) Offloads to default destinations, (b) Offloads to user’s home server, (c) Offloads to Load Balancer in user’s domain, (d) No offload

- User's remote IP address changes between log-in to load balancer server, and session initialization on offload server

  LON-CAPA expects the remote IP address of the client web request to be the same at the authentication stage on the load balancer, and then subsequently, following attempted session migration to the offload server. Where IPs do not match (e.g., a user is accessing LON-CAPA using an internet connection provided by a cellular network, where IP addresses are pooled and can change between requests), you can set how that situation should be handled. The options are:

  - Session hosted on Load Balancer, after re-authentication
  - Session hosted on offload server, after re-authentication
  - Session hosted on a specified server (selected from the domain's servers), after re-authentication

  If the load balancing server's domain and the domain being configured are the same, you will select one of these options separately for (a) Single Sign On (SSO) users from your domain, and also (b) for both non-SSO users from your domain, and for users from other domains in the LON-CAPA network (if you have joined a cluster). If your institution maintains multiple domains then a load balancer can have a different domain from the domain you are currently configuring. In that instance you will select one of these options separately for (a) SSO users from your domain and (b) non-SSO users from your domain.

## 2.21 Encrypting server traffic with SSL

There are two different contexts in which a LON-CAPA server may communicate via SSL (Secure Sockets Layer):

- Encrypted web pages served by Apache via port 443. In this case, client requests will be for URLs beginning https://.
- Encrypted internal communication between LON-CAPA servers via port 5663.

**Apache SSL**
In the case of Apache, the steps required depend on the Linux distro.

- CentOS/RedHat/Scientific Linux/Fedora:

  *yum install mod_ssl*

- SuSE/SLES:

  Check that ssl is included in the list of modules in the *APACHE_MODULES* string in */etc/sysconfig/apache2*.

- Debian/Ubuntu LTS:

*a2enmod ssl*

For all distros you will need to install a key, generate a certificate signing request with that key, and have the certificate signed. You will also want to disable the passphrase prompt on web server restart by removing the password from the copy of the key you use with Apache, e.g.,

*openssl rsa -in server.key -out server.key.nopass*

You will then put the the (nopass) key and certificate files in locations accessible to Apache, and include information about the locations of those files in a config file containing the following lines:

SSLCertificateFile <path to signed certificate>

SSLCertificateKeyFile <path to key>

replacing <path to ...> with the path to the location of the particular file.

Which Apache config file contains these entries depends on the distro:

- CentOS/RedHat/Scientific Linux/Fedora:

  /etc/httpd/conf.d/ssl.conf

- SuSE/SLES

  /etc/apache2/vhosts.d/vhost-ssl.conf

  (copied from vhost-ssl.conf with the entry for DocumentRoot changed to "/home/httpd/html").

- Debian/Ubuntu LTS

  /etc/apache2/sites-available/000-default-ssl

If you want to use rewrite rules to ensure that all external web requests are served using SSL, you should verify that mod_rewrite is enabled:

- CentOS/RedHat/Scientific Linux/Fedora

  Verify that the following entry in /etc/httpd/conf/httpd.conf is not commented out: *LoadModule rewrite_module modules/mod_rewrite.so*

- SuSE/SLES

  Check that *rewrite* is included in the list of modules in the *APACHE_MODULES* string in /etc/sysconfig/apache2.

- Debian/Ubuntu LTS

*a2enmod rewrite*

You will also need to copy the rewrites/loncapa_rewrite_on.conf file to loncapa_rewrite.conf with the following commands:

- CentOS/RedHat/Scientific Linux/Fedora

  *cp /etc/httpd/conf/rewrites/loncapa_rewrite_on.conf /etc/httpd/conf/loncapa_rewrite.conf*

- SuSE/SLES/Debian/Ubuntu LTS

  *cp /etc/apache2/rewrites/loncapa_rewrite_on.conf /etc/apache2/loncapa_rewrite.conf*

and then reload the web server:

- CentOS/RedHat/Scientific Linux/Fedora

  */etc/init.d/httpd reload*

- SuSE/SLES/Debian/Ubuntu LTS

  */etc/init.d/apache2 reload*

To disable rewriting of external web requests to https://, copy rewrites/loncapa_rewrite_off.conf to loncapa_rewrite.conf and reload the web server.

You will need to open the server's Firewall to allow inbound traffic on port 443.

- CentOS/RedHat/Scientific Linux

  */usr/bin/system-config-securitylevel-tui*

- Fedora

  */usr/bin/system-config-firewall-tui*

- SuSE/SLES

  yast -> Security and Users -> Firewall

- Debian 6/Ubuntu LTS

  *ufw allow 443/tcp*

Note: changing firewall settings will cause iptables to reload, which means the rules to allow connections from other LON-CAPA servers via port 5663 will need to be re-established (if the LON-CAPA daemons were already running) by doing: */etc/init.d/loncontrol restart* as root.

**Internal LON-CAPA SSL**

In the case of encrypted internal communication between LON-CAPA servers, you will need command line access as either root or www and enter the following commands:

*cd /home/httpd/lonCerts*

*sh request_ssl_key.sh*

**Important**: for the Common Name you should enter the lonHostID. This is displayed on the log-in page (Server: ) and is also an entry in the loncapa.conf file in /etc/httpd/conf (CentOS RedHat Scientific Linux Fedora) or /etc/apache2 (SuSE SLES Debian Ubuntu LTS). An example would be msul1.

By running *request_ssl_key.sh* you will:

- Generate a private/public key pair.

  The private key will be stored in /home/httpd/lonCerts/lonKey.pem It will be set so that only www can read this file. (You will want to make sure this file stays secret).

- Automatically send an e-mail to the LON-CAPA certificate authority. containing your public key so LON-CAPA can sign it.

Your certificate will be signed by the certificate authority and an e-mail will be sent to the e-mail address you gave when prompted for one when you ran request_ssl_key.sh.

Save the e-mail you receive to a file, remove the headers from it, and run it (as the *www* user).

If it successfully completes you will have:

- /home/httpd/lonCerts/lonhostcert.pem

  (your signed public key)

- /home/httpd/lonCerts/loncapaCA.pem

  (the public key of the Lon-CAPA certificate authority)

Now when you machine connects to another server in the LON-CAPA network it will try to do so over an SSL connection. You can verify this by doing:

*ps auxwww — grep lonc*

You should see something like:

lonc: msul1 Connection count: 1 Retries remaining: 5 (ssl)

where before you saw something like:

lonc: msul1 Connection count: 1 Retries remaining: 5 (insecure)

# 3 Domain Management

## 3.1 Creating Domain Coordinators

When LON-CAPA was first installed for a domain, an initial library server will have been set up, and the command

perl loncom/build/make_domain_coordinator.pl will have been run (as root) in the loncapa-X.Y.Z directory created when the LON-CAPA tarball was uncompressed.

This command will have created a new Linux user, who will be filesystem authenticated when logging into LON-CAPA, and who will have been assigned the Domain Coordinator role.

A Domain Coordinator can add users to the domain and assign any role in the domain with the exception of the Domain Coordinator role. To assign the Domain Coordinator role to other users, someone with root privileges on a library server in the domain can:

- run perl make_domain_coordinator.pl to create a new user (filesystem authenticated). make_domain_coordinator.pl will fail if:
  - the user already has a Linux account, or
  - the username is already in use for an existing LON-CAPA user in the domain.
- assign the Domain Coordinator role to an existing LON-CAPA user by running the following command in the loncapa-X.Y.Z directory
  - perl loncom/build/add_domain_coordinator_privilege.pl [USERNAME:DOMAIN] [DCDOMAIN]
    - where USERNAME:DOMAIN are the username and domain of an existing user, who is to be granted Domain Coordinator privileges,
    - DCDOMAIN is the domain to be coordinated. Note: DCDOMAIN must be a domain for which the server where the command is run is a library server. Furthermore, the home library server of the user to whom the role is to be assigned must be the server where the command is being run.

## 3.2 Revoking Domain Coordinator Roles

A Domain Coordinator can expire any role in the domain with the exception of the Domain Coordinator role. To expire a Domain Coordinator role for a user whose home server is the current server requires command line access, to the domain's library server(s).

When you install LON-CAPA you uncompress a tarfile to create a lon-capa-X.Y.Z directory where X, Y and Z are version numbers, e.g., 2.10.0. Within that directory, as root, run the script used to expire roles:

perl loncom/build/expire_DC_role.pl [USERNAME:DOMAIN] [DCDOMAIN]

- where USERNAME:DOMAIN are the username and domain of an existing user, for whom the Domain Coordinator privilege is to be revoked.

- DCDOMAIN is the domain for which the role is being revoked. Note: DCDOMAIN must be a domain for which the server is is a library server.

## 3.3 Scheduling of Scripts Run Periodically

When LON-CAPA is installed a file named loncapa is written to /etc/cron.d. The frequency and timing of execution of scripts included in this loncapa crontab file can be modified to suit the needs of your domain. The scripts, which are all run as the user 'www', are as follows:

- */home/httpd/perl/loncron* run daily at 5.10 am updates the list of servers in the LON-CAPA cluster to which your domain belongs. All servers in this list should be contactable, and will have the ability to host user sessions for users in the domain, as well as being available, if designated as library servers, to return responses to remote searches for files housed there which have been contributed to the LON-CAPA content repository. Connections to all servers are re-evaluated by loncron, in case some machines had become unavailable within the last 24 hours, and had therefore been flagged as temporarily offline.

- */home/httpd/perl/checkforupdates.pl* run daily at 4.10 am retrieves checksums/versions for LON-CAPA perl modules and perl script files for the version of LON-CAPA on your server, from the authoritative server(s) in the LON-CAPA cluster to which your domain belongs. If discrepancies are found (version and/or checksum) for installed files, or if files are missing, or extra (now obsolete) files are present this will be recorded in a log file, and also sent in an e-mail to recipients specified for the domain. The availability of a newer release of LON-CAPA on the install site – *http://install.loncapa.org/* – will also be checked, and notification will be icluded in the e-mail if LON-CAPA should be updated.

- */usr/local/loncapa/bin/CHECKRPMS* runs every other day at 3.10 am. This file automates the process of checking for available updates to LON-CAPA systems. The distprobe script, installed as a part of LON-CAPA, is used to determine the Linux distribution installed on the server, which in turn dictates which utility (yum, up2date, you rug, apt-get or zypper) is called to perform the package check.

- */home/httpd/perl/searchcat.pl* run every other day at 1.10 am traverses the LON-CAPA resource directory in a domain and gathers metadata which are entered into a SQL database. The script will repopulate and refresh the metadata database used for the searching the resource catalog. The script also refreshes and repopulates database tables used to store metadata for publicly accessible portfolio files, and user information needed for user searches in a LON-CAPA domain.

- */home/httpd/perl/cleanup_database.pl* run daily at 2.13 am drops tables from the LON-CAPA MySQL database if their comment is 'temporary' and they have not been modified in a given time (default is 2 days).

- */home/httpd/perl/refresh_courseids_db.pl* run daily at 2.50 am refreshes the database file (stored on a library server) queried when a fast lookup is needed for information about courses housed on the server. Course information includes the minimum LON-CAPA version needed to support the resources and/or settings used in the course.
- */home/httpd/perl/cleanup_file_caches.pl* run daily at 1.05 am removes temporary files from the LON-CAPA print spool, the multidownload zip spool, and userfiles cache.
- */home/httpd/perl/Autoenroll.pl* run daily at 1.30 am updates classlists for any LON-CAPA courses for which auto-enrollment is active, if enabled in the domain. A conduit needs to have been established to institutional course roster information.
- */home/httpd/perl/Autoupdate.pl* run daily at 3.30 am can reconcile first name, last name etc. information stored in LON-CAPA with authoritative data available from an institutional directory. A conduit needs to have been established to the institutional data source.
- */home/httpd/perl/Autocreate.pl* run daily at 2.30 am, checks for requests for creation of official courses, queued pending validation of instructor of record status. Also creates any pending course requests made using legacy web forms which store XML-based course descriptions in a "pending" directory.

## 3.4 Creating Courses

The "Course and Community Creation" menu provides access to a number of utilities which Domain Coordinators can use to either manage the process of course creation. An individual course or community can be created interactively by completing a web form which offers several choices for generating the new course or community:

- Username: set the username of the owner, who will also receive a coordinator role.
- Course Title: This is the name under which the course will appear on the Roles screen.
- Course Home Server: This is the server where the course will be housed.
- Course ID: This is an optional parameter to internally label the course. Stored in the course environment.
- Course Code: This is the institutional code used to identify the course according to the naming scheme adopted by your institution for "official" courses.
- Map: This the top level sequence file in the course. If left blank, a standard course will be created containing the first resource item (see below).
- NOT Standard Course: It this box is checked, the Map above becomes the top-level map for the course, and the main course sequence cannot be edited in DOCS. For standard courses, the top-level map is a course-specific "uploaded" document, and points to the above map.

- First Resource: This is the first resource which comes up after somebody selects a role in this course (standard courses only).
- Clone an existing course: An alternative to creation of a new course with a single item or a non-standard course with a sequence file as the top-level map is to copy the contents of an existing course into the new course.
- Open all assessments: Sets the course-level open date for all assignments to "now". Circumvents the "not open to be viewed" problem.
- Set policy/content feedback: Sets the recipient address for these types of feedback messages to the course coordinator.
- Disable student resource discussion: Disables the "bottom-of-the-page" discussions.

Two types of course "containers" are currently available in LON-CAPA:

- Course
- Community

The key difference between the two is that the Coordinator of a Course may import any published resource into the course as long as he/she has access rights for it, whereas import into a Community is restricted to just those resources for which the Coordinator is the author, or a co-author. In addition, the names of standard roles in the two containers have different names: (Course Coordinator vs. Coordinator, Instructor vs. Leader, Teaching Assistant vs. Assistant Leader, Student vs. Member).

In addition to using a web form to create courses one-at-a-time, Domain Coordinators can upload an XML file containing descriptions of courses (see: **??** Batch Creation) to create multiple courses. If the ability to request courses has been enabled, and certain course types have been set to require Domain Coordinator approval, then the "Approve or reject requests" item can be used to display a list of requests requiring approval. These may be approved or rejected. If a conduit has been established to an institutional data source which provides information about instructors of record, then the "View pending official course requests" may contain a list of requests for official courses, held pending validation. Validation can be attempted; if it still fails it is possible to override this and force creation of the course.

Lastly, there is access to a utility which can be used to display course and community creation history.

# 4 Integration with Institutional Systems

## 4.1 Institutional Authentication (non-SSO)

When a user is assigned an authentication type of "Local authentication" , the perl module /home/httpd/lib/perl/localauth.pm will be used to evaluate the user's credentials. The documentation included in the stub provided with a LON-CAPA installation describes the basic operation of localauth.pm

The localauth routine receives four arguments (in the order: two required, one optrional, another required).

1. the username the user types in.
2. the password the user typed in.
3. optional information stored when the authentication mechanism was specified for the user ("Local authentication with argument: ....")
4. the domain the user typed in.

The routine will return 1 if the user is authenticated and 0 otherwise, and it can optionally return a negative value for an error condition. This negative value will be logged along with the username used in the failed authentication which resulted in the error condition.

A common use of localauth.pm is to connect with an LDAP service.

```
package localauth;
use strict;
use Net::LDAP;
use Net::LDAPS;
sub localauth {
    my ($username,$password) = @_;
    my $ldap_host_name = ''; # insert the host name of your ldap
 server, e.g., ldap.msu.edu
    my $ldap_ca_file_name = ''; # insert the ldap certificate
 filename - include absolute path
    # certificate is required if you wish to encrypt the password.
    # e.g., /home/http/perl/lib/local/ldap.certificate
    my $ldap_search_base = ''; # ldap search base, this might
 be set to 'o=msu.edu'.
    my $ldap = Net::LDAPS->new(
        $ldap_host_name,
        verify => 'require', # 'require' -> a certificate
     is needed, -> 'none' if no certificate used
        cafile => $ldap_ca_file_name,
 );
    if (!(defined($ldap))) {
        return (0);
 }
    $ldap->bind;
    my $search_string = '(uid='.$username.')';
    my $mesg = $ldap->search (
        base => $ldap_search_base,
        filter => $search_string,
        attrs => ['dn'] ,
```

```
        );
           if ($mesg->code) {
               $ldap->unbind;
               $ldap->disconnect;
               return (0);
        }
           my @entries = $mesg->all_entries;
           if (@entries > 0) {
               $ldap->unbind;
               $ldap->disconnect;
               return (0);
        }
           $mesg = $ldap->bind (
               dn => $entries[0]->dn,
               password => $password,
        );
           $ldap->unbind;
           $ldap->disconnect;
           if ($mesg->code) {
               return (0)
        }
           return (1);
    }
       1;
```

## 4.2 Shibboleth Authentication (SSO)

If your institution operates a Shibboleth Identity Provider (IdP) for your users, then you can configure a LON-CAPA server to authenticate users by running your server as a Shibboleth Service Provider (SP).

To configure a LON-CAPA server as a Shibboleth SP you will need to:

- Install Shibboleth packages for your Linux distro, or build/install from source
- Modify your Apache configuration to include shib.conf (which will load mod_shib)
- Set shibd to start on boot
- Install mod_ssl and Apache/SSL certificates
- Configure your SP to work with your institution's IdP
- Add a custom Apache config file to include some PerlVars (for logout etc.)

Although Shibboleth can be built on any 32 or 64 bit Linux distro on which LONCAPA is supported, official packages are available from http://shibboleth.net for: Red Hat/CentOS 5, 6 and 7, SLES 10 and 11, and openSuSE 12.1, 12.2, and 12.3. In addition, http://www.switch.ch provides a repository from which shibboleth packages may be obtained for Ubuntu 12.04 LTS and 14.04 LTS.

1. Install Shibboleth

   See: https://wiki.shibboleth.net/confluence/display/SHIB2/NativeSPLinuxInstall

   Shibboleth repos for RPM-based Linux distros can be found at:

   http://download.opensuse.org/repositories/security:/shibboleth/

   Red Hat/CentOS – add shibboleth.repo to /etc/yum.repos.d

   e.g., CentOS 5

```
[security_shibboleth]
name=Shibboleth (CentOS_5)
type=rpm-md
baseurl=http://download.opensuse.org/repositories/security:/shibboleth/CentOS_5/
gpgcheck=1
gpgkey=http://download.opensuse.org/repositories/security:/shibboleth/CentOS_5/
repodata/repomd.xml.key
enabled=1
```

   e.g., CentOS 6

```
[security_shibboleth]
name=Shibboleth (CentOS_6)
type=rpm-md
baseurl=http://download.opensuse.org/repositories/security:/shibboleth/
CentOS_CentOS-6/
gpgcheck=1
gpgkey=http://download.opensuse.org/repositories/security:/shibboleth/
CentOS_CentOS-6/repodata/repomd.xml.key
enabled=1
```

   Then do:

```
yum install shibboleth
```

   SLES/openSuSE:

   e.g. SLES 11 SP3:

```
zypper addrepo http://download.opensuse.org/repositories/security:shibboleth/
SLE_11_SP3/security:shibboleth.repo
zypper refresh
zypper install shibboleth
```

e.g. SuSE 12.3

```
zypper addrepo http://download.opensuse.org/repositories/security:shibboleth/
openSUSE_12.3/security:shibboleth.repo
zypper refresh
zypper install shibboleth
```

e.g., Ubuntu 12.04LTS

See: https://www.switch.ch/aai/docs/shibboleth/SWITCH/2.5/sp/deployment/?os=ubuntu

```
sudo apt-get install curl
sudo curl -k -O http://pkg.switch.ch/switchaai/SWITCHaai-swdistrib.asc
sudo apt-key add SWITCHaai-swdistrib.asc
echo 'deb http://pkg.switch.ch/switchaai/ubuntu precise main' |
sudo tee /etc/apt/sources.list.d/SWITCHaai-swdistrib.list $>$ /dev/null
sudo apt-get update
sudo apt-get install shibboleth
```

The following directories will have now been created:

/etc/shibboleth

/var/log/shibboleth

/var/run/shibboleth

/var/cache/shibboleth

2. Apache configuration

   Red Hat/CentOS – httpd.conf should be modified to contain:

   ```
   UseCanonicalName On
   Include conf/shib.conf
   ```

   The Include should precede the line: Include conf/loncapa_apache.conf

   SLES/SuSE – modify /etc/apache2/default-server.conf and /etc/sysconfig/apache2

   Modify /etc/sysconfig/apache2 to include:

   ```
   APACHE_USE_CANONICAL_NAME=``on''
   ```

   Modify /etc/apache2/default-server.conf to contain:

   ```
   Include conf/shib.conf
   ```

The Include should precede the line: Include conf/loncapa_apache.conf

Note: the shib.conf file should include: ShibUseHeaders Off so that environment variables can be used to access user attributes, if needed.

3. Set shibd to start on boot

```
/sbin/chkconfig shibd on
```

4. Install mod_ssl and Apache/SSL certificates

   (a) Red Hat/CentOS

   ```
   yum install mod_ssl
   ```

   (b) SuSE/SLES

   Use yast -> Network Services -> HTTP Server -> Server Modules to set ssl to enabled, and rewrite to enabled

   (c) Ubuntu

   ```
   sudo a2enmod ssl
   sudo a2enmod rewrite
   sudo a2ensite default-ssl.conf
   ```

   Edit default-ssl.conf replace "DocumentRoot /var/www/html" with "DocumentRoot /home/httpd/html"

   You also need to create an SSL certificate signing request and have it signed by a certificate authority, before installing the signed certificate and the corresponding key. Also you will need to open port 443 in the firewall, and enable rewrites of http to https (see "Encrypting server traffic with SSL" 2.21 section for more information).

5. Configure your SP to work with your institution's IdP

   Before customizing your Shibboleth SP to work with your IdP it is recommended to test the default configuration.

   /usr/sbin/shibd -t should report:

   overall configuration is loadable, check console for non-fatal problems

   After restarting your Apache server, the result of accessing the URL:

   https://<yourserver.edu>/Shibboleth.sso/Session

   should be: "A valid session was not found."

   A Shibboleth SP can also be tested with http://testshib.org/

   Once preliminary testing is complete you will need to edit /etc/shibboleth/shibboleth2.xml based on information provided by your institution, for your SP to work with the appropriate IdP. At a minimum the shibboleth2.xml file will need to contain the service hostname of your LON-CAPA server, and the SP entityID. You should also set an e-mail address, for users to contact in case of errors.

   e.g.,

```
<SPConfig xmlns="urn:mace:shibboleth:2.0:native:sp:config"
    xmlns:conf="urn:mace:shibboleth:2.0:native:sp:config"
    xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
    xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
    xmlns:md="urn:oasis:names:tc:SAML:2.0:metadata"
    logger="syslog.logger" clockSkew="180">

    <InProcess logger="native.logger">
        <ISAPI normalizeRequest="true" safeHeaderNames="true">
            <Site id="1" name="yourserver.someplace.edu"/>
        </ISAPI>
    </InProcess>

    <Host name="yourserver.someplace.edu">
        <Path name="secure" authType="shibboleth" requireSession="true"/>
    </Host>

    <ApplicationDefaults id="default" policyId="default"
        entityID="https://yourserver.someplace.edu/shibboleth"
        REMOTE_USER="eppn persistent-id targeted-id"
        signing="false" encryption="false">

        <Sessions lifetime="28800" timeout="3600" checkAddress="false"
            handlerURL="/Shibboleth.sso" handlerSSL="true"

        </Sessions>

        <Errors supportContact="helpdesk@someplace.edu"
            logoLocation="/shibboleth-sp/logo.jpg"
            styleSheet="/shibboleth-sp/main.css"/>
    </ApplicationDefaults>
</SPConfig>
```

REMOTE_USER is used to pass on the primary identifier of the authenticated user. It should be set to match an attribute or alias defined in the attribute-map.xml file. LON-CAPA uses this value ($r->user in the mod_perl environment) as the username of the user. The user's domain will be either the value of the PerlVar lonSSOUserDomain, or if that is undefined, the PerlVar lonDefDomain. If the attribute used for REMOTE_USER is in the form: username@somewhere.edu, and somewhere.edu is the "internet domain" (i.e., the last item in the colon separated list of entries for your server in /home/httpd/lonTabs/hosts.tab), then LON-CAPA will automatically remove the @somewhere.edu, such that $r->user will be just username.

6. Add a custom Apache config file to include some PerlVars (for logout etc.)

   Add a file to your Apache conf directory named loncapa_apache_local<dom>.conf, where <dom> is domain, to include items such as:

```
PerlSetVar lonSSOUserLogoutHeadFile_<dom>/home/httpd/html/adm/sso_logout_head
PerlSetVar lonSSOUserLogoutMessageFile_<dom> /home/httpd/html/adm/sso_logout_body
PerlSetVar lonSSOUserUnknownRedirect /adm/sso_failed_login.html
PerlSetVar lonSSOUserDomain <dom>
```

   and add the corresponding files owned by www:www in /home/httpd/html/adm/

   Notes:

   (a) All files will contain HTML mark-up, but the sso_logout_head item is a fragment inserted into the head block of the standard LON-CAPA logout page, and similarly, the sso_logout_body is a fragment inserted into the body of the page, whereas the sso_failed_login.html file should be a complete HTML document.

   If the name of the PerlVar ends _<dom> then the HTML fragment is only displayed to SSO users from that particular domain. It is possible that a LON-CAPA user from another domain might have used SSO authentication on a server in his/her home domain, and then switched session to your server, (e.g., for co-author access to an Authoring Space in your domain). In that particular case, if you wanted to display custom HTML, you should add a PerlVar with a name ending in _<otherdom>. If you include PerlVars for lonSSOUserLogoutHeadFile and/or lonSSOUserLogoutMessageFile they will be included for SSO users who use the Logout link on your LON-CAPA regardless of the user's domain.

   (b) SAML 2 Single Logout (SLO) has limited support starting with IdP's running Shibboleth 2.4. The <Logout> element is used to enable and configure support for Logout protocols and behavior within the SP, e.g.,

```
<Logout>SAML2 Local</Logout>
```

   to support both local, i.e., for the SP itself (Local), and also in a limited way for the IdP (SAML2). In pre-2.4 Shibboleth2 /etc/shibboleth2.xml LogoutInitiators enable SP-initiated local logout e.g., https://yourserver/Shibboleth.sso/Logout.

   Depending on the availability of SLO support from your institution's IdP you should craft an appropriate message to include in sso_logout_body. If you include a link to the URL for a local logout, you should indicate that access to other web applications using SSO may continue to be available, even after logout from the specific SP. If no local logout is provided, then after logout from LON-CAPA, the web browser needs to be quit, to ensure access to LON-CAPA requires re-authentication.

   (c) If you enable self-creation of SSO-authenticated users, then the sso_failed_login.html document need not be created.

Attributes provided to the SP by the IdP are available to LON-CAPA as Environment variables. For Shibboleth SSO users, mapping of Shibboleth environment variable names to user data fields can be set so that the appropriate user information is available at account creation time. The mapping of variable name to LON-CAPA data name will be set by a domain cordinator using the domain configuration screen for "Users self-creating accounts".

Note: user data for a new user need not come from Environment variables populated by Shibboeth; instead it can come from a customized get_userinfo() routine in /home/httpd/lib/perl/localenroll.pm (see Directory Information 4.6 section).

e.g., sso_logout_body

```
<p>
As your original log-in to LON-CAPA was authenticated by a central Shibboleth
Single Sign On service, your Shibboleth credentials are still valid.<br />
Until you close your web browser, web applications which support Shibboleth
Single Sign-on (including LON-CAPA) will not require you to re-enter your
username and password</p>
<p>
To expire your active Shibboleth authentication token you must quit your web
browser.
</p>
```

e.g., sso_failed_login.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">

<head>
<meta http-equiv="Content-Type" content="text/html;charset=utf-8" />
<title>No LON-CAPA Account</title>
</head>
<body bgcolor="#ffffff">
<p>
You have authentication using Shibboleth Single Sign On service was
successful.<br />
However, you do not currently have a LON-CAPA account with the username
with which you authenticated.<br />
Policies at your institution do not allow you to create a LON-CAPA account
yourself, after successful authentication.
```

```
Please contact the <a href="/adm/helpdesk">LON-CAPA Helpdesk</a> for your
domain.
</p>
<p>
Your Shibboleth credentials are still valid.<br />
Until you close your web browser, web applications which support Shibboleth
Single Sign-on will not require you to re-enter your username and password
</p>
<p>
To expire your active Shibboleth authentication token you must quit your web
browser.
</p>
</body>
</html>
```

## 4.3 CAS Authentication (SSO)

The procedure for enabling institutional Single Sign On (SSO) via a central authentication service (CAS) that is not Shibboleth involves building or installing an Apache module provided by you institution, and then modifying an Apache configuration file on your LON-CAPA server to (a) load the module, and (b) configure LON-CAPA to use it, by default, when unauthenticated users access /adm/roles.

If your server will be part of the cluster of collaborating institutions, it is possible that users from other LON-CAPA domains might visit your server to log-in to LON-CAPA. To support that possibility, it is recommended that the CAS log-in page includes a link to point back at /adm/login on your LON-CAPA server, and the link is identified as one to be followed by users from other domains. See: https://loncapa.msu.edu/adm/roles for an example.

In order for Apache to use your CAS system you need to set the PerlVar lonOtherAuthen to yes, and provide the default domain for SSO users and the authentication type (i.e., the name of your CAS).

- Add a custom Apache config file to include some required PerlVars and load the CAS shared object.

  ```
  PerlSetVar lonOtherAuthen yes
  PerlSetVar lonOtherAuthenType MyCAS
  PerlSetVar lonSSOUserDomain <dom>

  LoadModule mod_sentinel modules/mod_mycas.so
  ```

  where <dom> is your domain, and mod_mycas.so is ths name of the CAS shared object. You might put the config file (mycas.conf) in: /etc/httpd/conf.d/ (CentOS/Red Hat/Scientific Linux), or in /etc/apache2/conf.d/ (SuSE/SLES) or /etc/apache2/conf-available (Ubuntu, and enabled with: sudo a2enconf).

- Add a custom Apache config file to include some optional PerlVars (for logout etc.)

  Add a file to your Apache conf directory named loncapa_apache_local<dom>.conf, where <dom> is domain, to include items such as:

  ```
  PerlSetVar lonSSOUserLogoutHeadFile_<dom> /home/httpd/html/adm/sso_logout_head
  PerlSetVar lonSSOUserLogoutMessageFile_<dom> /home/httpd/html/adm/sso_logout_body
  PerlSetVar lonSSOUserUnknownRedirect /adm/sso_failed_login.html
  PerlSetVar lonSSOReloginServer https://somehost.somewhere.edu
  ```

  and add the corresponding files owned by www:www in /home/httpd/html/adm/

Notes:

1. All files will contain HTML mark-up, but the sso_logout_head item is a fragment inserted into the head block of the standard LON-CAPA logout page, and similarly, the sso_logout_body is a fragment inserted into the body of the page, whereas the sso_failed_login.html file should be a complete HTML document.

   If the name of the PerlVar ends _<dom> then the HTML fragment is only displayed to SSO users from that particular domain. It is possible that a LON-CAPA user from another domain might have used SSO authentication on a server in his/her home domain, and then switched session to your server, (e.g., for co-author access to an Authoring Space in your domain). In that particular case, if you wanted to display custom HTML, you should add a PerlVar with a name ending in _<otherdom>. If you include PerlVars for lonSSOUserLogoutHeadFile and/or lonSSOUserLogoutMessageFile they will be included for SSO users who use the Logout link on your LON-CAPA regardless of the user's domain.

2. If you enable self-creation of SSO-authenticated users, then the sso_failed_login.html document need not be created.

3. If you would like the log-in again link on the logout page to point to a specific URL just for SSO users, then you would set the PerlVar for lonSSOReloginServer. However, if you would like the log-in link for all users from your domain (both SSO and non-SSO authenticated) to point at a particular URL, then you would log-in to LON-CAPA, select a Domain Coordinator role, and use Main Menu -> Set domain configuration -> Display ("Default authentication/language/timezone/portal/types" checked) an set the URL in "Portal/Default URL".

## 4.4 Institutional User Categories/Affiliations

Users in a domain can be assigned one or more institutional affiliations by the Autoupdate process which reconciles user information in LON-CAPA with institutional directory information. User type (or affiliation) can determine such things as (a) default portfolio quota, (b) the types of user information which may be updated in different contexts, (c) whether a user can self-enroll in a course.

Prior to LON-CAPA 2.11.0 the possible institutional types in a domain were defined by *inst_usertypes()*. Examples of institutional types might be: Faculty, Adjunct, Staff, Student etc. In addition to any types defined in *inst_usertypes()*, a type "other" will also be available for assignment to users who do not fall in any of the recognized categories of user. In the absence of any defined user categories, the type "other" applies to all users from a domain.

Starting with LON-CAPA 2.11.0 use of the inst_usertypes() subroutine is deprecated. The domain configuration web GUI accessible to Domain Coordinators is now used to manage institutional types. If you have previously customized the inst_usertypes() routine, then values set there will be used when displaying the "Institutional user types" section in the domain config screen for: Default authentication, language, timezone, portal and types.

Once a Domain Coordinator has visited that screen and saved the settings, configuration thereafter will be via the web GUI of values stored in the domain's configuration.db file on the primary library server in the domain, and values in inst_usertypes() will no longer be consulted. However, if you have created other custom routines in localenroll.pm which call inst_usertypes() internally, you will likely want to continue to maintain it.

### inst_usertypes

The routine accepted three arguments:

1. $dom - domain
2. $usertypes - reference to hash which will contain key = value, where keys are institution affiliation types (e.g., Faculty, Student etc.) and values are titles (e.g., Faculty/Academic Staff)
3. $order - reference to array which will contain the order in which institutional types should be shown when displaying data tables (e.g., default quotas or updateable user fields (see Domain Configuration menu)

The routine returns 'ok' if no errors occurred.

At MSU there are six different categories of users.

```
sub inst_usertypes {
    my ($dom,$usertypes,$order) = @_;
    my $outcome = 'ok';
    %{$usertypes} = (
        Faculty => 'Faculty/Academic Staff',
        Staff => 'Support Staff',
        Student => 'Student',
        Assistant => 'Assistant',
        StaffAff => 'Affiliate',
        StuAff => 'Student Affiliate'
  );
    @{$order}=('Faculty','Staff','Student','Assistant','StaffAff','StuAff');
    return $outcome;
}
```

## 4.5 Format Rule Definitions and Checks: Usernames and IDs

Format restrictions for usernames and student/employeeIDs for an institution, and formats which may *not* be used for e-mail addresses used as usernames when users self-create accounts are defined in three subroutines in localenroll.pm: *username_rules()*, *id_rules()*, and *selfcreate_rules()*. The three routines accept a similar set of arguments, and return 'ok' in each case, if no error occurred.

**username_rules** - Incoming data: three arguments

1. $dom - domain
2. $ruleshash - reference to hash containing rules (a hash of a hash)

   keys of top level hash are short names (e.g., netid, noncredit); for each key, value is a hash.

   - desc => long name for rule
   - rule => description of rule
   - authtype => (krb5,krb4,int, or loc) authentication type for rule
   - authparm => authentication parameter for rule
   - authparmfixed => 1 if authparm used when creating user for rule must be authparm
   - authmsg => Message to display describing authentication to use for this rule
3. $rulesorder - reference to array containing rule names in order to be displayed

At MSU, a NetID consists of eight characters or less, and will be authenticated by Kerberos (version 5) in the MSU.EDU realm. The rule itself is defined in *username_rules()*, and the code which checks for compliance is in *username_check()*:

```
sub username_rules {
    my ($dom,$ruleshash,$rulesorder) = @_;
    %{$ruleshash} = (
        netid => {
            name => 'MSU NetID',
            desc => 'Eight characters or less',
            authtype => 'krb5',
            authparm => 'MSU.EDU',
            authparmfixed => '',
            authmsg => 'A new user with a username which
        matches a valid MSU NetID will log-in using the
        MSU Net ID and MSU Net password.',
    }
```

```
        );
           @{$rulesorder} = ('netid');
           return 'ok';
    }
```

**id_rules** - Incoming data: three arguments

1. $dom - domain
2. $ruleshash - reference to hash containing rules (a hash of a hash)keys of top level hash are short names (e.g., studentID, employeeID); for each key, value is a hash
   - desc => long name for rule
   - rule => description of rule
3. $rulesorder - reference to array containing rule names in order to be displayed

At MSU, student/employee IDs are eight digits prefaced by A or Z. The rule itself is defined in *id_rules()*, and the code which checks for compliance is in *id_check()*:

```
    sub id_rules {
        my ($dom,$ruleshash,$rulesorder) = @_;
        %{$ruleshash} = (
            studentID => {
               name => 'MSU student PID',
               desc => 'Letter A or a, followed by eight digits',
         },
            facstaffID => {
               name => 'MSU faculty/staff ID',
               desc => 'Letter Z or z, followed by eight digits',
         },
        );
           @{$rulesorder} = ('studentID','facstaffID');
           return 'ok';
    }
```

**selfcreate_rules** - Incoming data: three arguments

1. $dom - domain
2. $ruleshash - reference to hash containing rules (a hash of a hash)

   keys of top level hash are short names (e.g., msuemail); for each key, value is a hash
   - desc => long name for rule

- rule => description of rule

3. $rulesorder - reference to array containing rule names in order to be displayed

At MSU all users receive a Net ID (e.g., *sparty*), and a corresponding e-mail account: *sparty@msu.edu*. So, at MSU the rules for e-mail addresses to be used as LON-CAPA usernames prohibit e-mails such as *sparty@msu.edu*. In such cases, the user should log-in with the sparty Net ID/password and request account creation for the username: *sparty*. The rule itself is defined in *selfcreate_rules()*, and the code which checks for compliance is in *selfcreate_check()*:

```
sub selfcreate_rules {
    my ($dom,$ruleshash,$rulesorder) = @_;
    %{$ruleshash} = (
        msuemail => {
        name => 'MSU e-mail address ',
        desc => 'netid@msu.edu',
  },
    );
    @{$rulesorder} = ('msuemail');
    return 'ok';
}
```

The corresponding routines which check for compliance with rules enabled via Domain Configuration-> User Creation are *username_check()*, *id_check()*, and *selfcreate_check()*. The three routines accept a similar set of four arguments, and return 'ok' in each case, if no error occurred.

1. $dom - domain (scalar)
2. $uname (username_check()), $id (id_check()) or $selfcreatename (selfcreate_check())
   - proposed username, id or self-created username being compared against rules (scalar)
3. $to_check (reference to array of rule names to check)
4. $resultshash (reference to hash of results) hash of results for rule checked
   keys are rule names - values are: 1 or 0 (for matched or unmatched)

The routines used for checking rule compliance at MSU are as follows:

## username_check

```
sub username_check {
    my ($dom,$uname,$to_check,$resultshash) = @_;
    my $outcome;
    if (ref($to_check) eq 'ARRAY') {
```

```
            foreach my $item (@{$to_check}) {
                if ($item eq 'netid') {
                  if ($uname =~ /^\w{2,8}$/) {
                    $resultshash->{$item} = 1;
                } else {
                    $resultshash->{$item} = 0;
                }
          }
            $outcome = 'ok';
       }
          return $outcome;
    }
```

### id_check

```
      sub id_check {
          my ($dom,$id,$to_check,$resultshash) = @_;
          my $outcome;
          if (ref($to_check) eq 'ARRAY') {
            foreach my $item (@{$to_check}) {
                if ($item eq 'facstaffID') {
                  if ($id =~ /^z\d{8}$/i) {
                    $resultshash->{$item} = 1;
                } else {
                    $resultshash->{$item} = 0;
                }
            } elsif ($item eq 'studentID') {
                  if ($id =~ /^a\d{8}$/i) {
                    $resultshash->{$item} = 1;
                } else {
                    $resultshash->{$item} = 0;
                }
            }
          }
            $outcome = 'ok';
       }
          return $outcome;
    }
```

### selfcreate_check

```
sub selfcreate_check {
    my ($dom,$selfcreatename,$to_check,$resultshash) = @_;
    my $outcome;
    if (ref($to_check) eq 'ARRAY') {
        foreach my $item (@{$to_check}) {
            if ($item eq 'msuemail') {
              if ($selfcreatename =~ /^\w{2,8}\@msu\.edu$/)
            {
                $resultshash->{$item} = 1;
            } else {
                $resultshash->{$item} = 0;
            }
        }
    }
        $outcome = 'ok';
  }
    return $outcome;
}
```

## 4.6 Institutional Directory Information

Two subroutines exist in localenroll.pm to provide a connection between institutional directory data (e.g., user information from LDAP) and LON-CAPA. The first is *get_userinfo()* which can operate in two modes.: (a) it can be used to provide first name, last name, e-mail address, student/employee ID etc., for a specified username, e.g., for a new user being created in LON-CAPA, and (b) it can be used to retrieve user information for multiple users from an institutional directory searches when (for example) a course coordinator is adding a new user directly to a course. At MSU the routine which actually queries institutional data sources is itself called by *get_userinfo()*. This was done so that the same underlying routine can also be used by the second of the two subroutines: *allusers_info()* which is called by Autoupdate.pl (a script which can be run periodically to reconcile user information in LON-CAPA with institutional directory data for all users).

### get_userinfo

Four required arguments and additional optional arguments
Two modes of operation:

1. Retrieves institutional data for a single user either by username, if $uname is included as second argument,

   or by ID if $id is included as a third argument. Either second or third arguments must be provided; seventh, eighth and ninth args will be undefined.

2. Retrieves institutional user data from search of an institutional directory based on a search. seventh and eighth args are required; ninth is optional. second and third will be undefined.

Arguments:

1. $dom - domain
2. $uname - username of user
3. $id - student/faculty ID of user
4. $instusers - reference to hash which will contain info for user as key = value; keys will be one or all of: lastname, firstname, middlename, generation, id, inststatus - institutional status (e.g., faculty,staff,student).

   Values are all scalars except inststatus, which is an array.
5. $instids - reference to hash which will contain ID numbers - keys will be unique IDs (student or faculty/staff ID)

   values will be either: scalar (username) or an array if a single ID matches multiple usernames.
6. $types - optional reference to array which contains institutional types to check.
7. $srchby - optional if $uname or $id defined, otherwise required.

   Allowed values include: 1. lastfirst, 2. last, 3. uname corresponding to searches by 1. lastname,firstname; 2. lastname; 3. username
8. $srchterm - optional if $uname or $id defined, otherwise required - String to search for.
9. $srchtype - optional. Allowed values: contains, begins (defaults to exact match otherwise).

Returns 'ok' if no error occurred. Side effects - populates the $instusers and $instids refs to hashes with information for specified username, or specified id, if fifth argument provided, from all available, or specified (e.g., faculty only) institutional datafeeds, if sixth argument provided.

At MSU six separate MS-SQL database tables are queried, with each table corresponding to a specific institutional type. A routine is called to connect to the database. and the actual queries are handled by a separate routine - *query_user_tables()*.

```
sub get_userinfo {
    my ($dom,$uname,$id,$instusers,$instids,$types,
        $srchby,$srchterm,$srchtype) = @_;
 my $outcome;
        my @srchtables;
```

```
    my %tables = (
          Faculty => 'FACULTY_VU',
          Staff => 'STAFF_VU',
          Student => 'STUDENT',
          Assistant => 'ASSISTANT',
          StaffAff => 'AFFILIATE',
          StuAff => 'STUDENT_AFFILIATE'
    );
       my ($dbh,$dbflag) = &connect_DB('HR');
       foreach my $type (@{$types}) {
          if (exists($tables{$type})) {
             push(@srchtables,$tables{$type});
       }
    }
       if (@srchtables == 0) {
          foreach my $type (keys(%tables)) {
             push(@srchtables,$tables{$type});
       }
    }
       if ($srchby eq '' && $srchterm eq '') {
          if ($uname ne '') {
             $srchby = 'uname';
             $srchterm = $uname;
       } elsif ($id ne '') {
             $srchby = 'id';
             $srchterm = $id;
       }
    }
       if ($srchterm ne '') {
          $outcome = &query_user_tables($dbflag,$dbh,\@srchtables,
                     $instusers,$instids,$srchby,$srchterm,$srchtype,$types);
    }
       if ($dbflag) {
          &disconnect_DB($dbh);
    }
       return $outcome;
}
```

Although query_user_tables() is not a subroutine included as a stub in the standard localenroll.pm, it is included below to show how the database queries are implemented at MSU.

```
sub query_user_tables {
```

```
   my ($dbflag,$dbh,$srchtables,$instusers,$instids,$srchby,$srchterm,
$srchtype,$types) = @_;
   my ($outcome,$condition,%multipids,$ldapfilter);
   if ($srchby eq 'uname') {
       if ($srchterm =~ /^\w{2,8}$/) {
          if ($srchtype eq 'contains') {
            $condition = "WHERE MSUNetID LIKE '%$srchterm%'";
            $ldapfilter = '(uid=*'.$srchterm.'*)';
          } elsif ($srchtype eq 'begins') {
            $condition = "WHERE MSUNetID LIKE '$srchterm%'";
            $ldapfilter = '(uid='.$srchterm.'*)';
          } else {
            $condition = "WHERE MSUNetID = '$srchterm'";
            $ldapfilter = '(uid='.$srchterm.')';
          }
    }
} elsif ($srchby eq 'lastname') {
       if ($srchterm =~ /[A-Za-z\-\.'\s]+/) {
          if ($srchtype eq 'contains') {
            if ($dbflag) {
              my $quoted_last = $dbh->quote('%'.$srchterm.'%');
              $condition = "WHERE LastName LIKE $quoted_last";
            }
            $ldapfilter = '(sn=*'.$srchterm.'*)';
          } elsif ($srchtype eq 'begins') {
            if ($dbflag) {
              my $quoted_last = $dbh->quote($srchterm.'%');
              $condition = "WHERE LastName LIKE $quoted_last";
            }
            $ldapfilter = '(sn='.$srchterm.'*)';
          } else {
            if ($dbflag) {
             my $quoted_last = $dbh->quote($srchterm);
              $condition = "WHERE LastName = $quoted_last";
            }
            $ldapfilter = '(sn='.$srchterm.')';
          }
    }
} elsif ($srchby eq 'lastfirst') {
       my ($srchlast,$srchfirst) = split(/,/,$srchterm);
       $srchlast =~ s/\s+$//;
       $srchfirst =~ s/^\s+//;
```

```
            if (($srchlast =~ /[A-Za-z\-\.'\s]+/) && ($srchfirst
        =~ /[A-Za-z\-\.'\s]+/)) {
               my ($quoted_first,$quoted_last);
               if ($srchtype eq 'contains') {
                 if ($dbflag) {
                   $quoted_last = $dbh->quote('%'.$srchlast.'%');
                   $quoted_first = $dbh->quote('%'.$srchfirst.'%');
                   $condition = "WHERE ( LastName LIKE $quoted_last
            AND FirstName LIKE $quoted_first )";
                 }
                 $ldapfilter = '(&(sn='.$srchlast.'*)(givenName='.$srchfirst.'*))';
               } else {
                 foreach my $table (@{$srchtables}) {
                   next if ($srchby && $condition eq '');
                   my $statement = "SELECT MSUNetID,Pid,FirstName,LastName,
                Person_Type FROM $table $condition";
                   my $sth = $dbh->prepare("$statement");
                   $sth->execute();
                   while ( my($uname,$pid,$first,$last,$type)
                = $sth->fetchrow_array ) {
                      $pid=lc($pid);
                      if (ref($instusers->{$uname}) eq 'HASH')
                {
                         if (ref($instusers->{$uname}{'instst
              if ($dbflag) {
                   $quoted_last = $dbh->quote($srchterm);
                   $quoted_first = $dbh->quote($srchterm);
                   $condition = "WHERE ( LastName = $quoted_last
            AND FirstName = $quoted_first )";
                 }
                 $ldapfilter = '(&(sn='.$srchlast.')(givenName='.$srchfirst.'))';
               }
        }
     } elsif ($srchby eq 'id') {
            if ($dbflag) {
              if ($srchterm =~ /^[AZ]\d{8}$/) {
                 $condition = "WHERE Pid = '$srchterm'";
           }
            }
     }
        if ($dbflag) {
            foreach my $table (@{$srchtables}) {
```

```
   next if ($srchby && $condition eq '');
   my $statement = "SELECT MSUNetID,Pid,FirstName,LastName,Person_Type
FROM $table $condition";
   my $sth = $dbh->prepare("$statement");
   $sth->execute();
   while ( my($uname,$pid,$first,$last,$type)
= $sth->fetchrow_array ) {
     $pid=lc($pid);
     if (ref($instusers->{$uname}) eq 'HASH')
{
       if (ref($instusers->{$uname}{'inststatus'})
eq 'ARRAY') {
         if (!grep(/^$type$/,@{$instusers->{$uname}{'inststatus'}}))
{
           push(@{$instusers->{$uname}{'inststatus'}},$type);
         }
       }
       if ($pid ne $instusers->{$uname}{'id'})
{
         if ($instusers->{$uname}{'id'} =~ /^A\d{8}$/)
{
           if ($pid =~ /^A\d{8}$/) {
             if (ref($multipids{$uname}) eq 'ARRAY')
{
               if (!grep(/^$pid$/,@{$multipids{$uname}}))
{
                 push(@{$multipids{$uname}},$pid);
               }
             } else {
               @{$multipids{$uname}}=($instusers->{$uname}{'id'},$pid);
             }
             $instusers->{$uname}{'id'} = $pid;
           }
         } elsif ($instusers->{$uname}{'id'} =~
/^Z\d{8}$/) {
           if ($pid =~ /^Z\d{8}$/) {
             if (ref($multipids{$uname}) eq 'ARRAY')
{
               if (!grep(/^$pid$/,@{$multipids{$uname}}))
{
                 push(@{$multipids{$uname}},$pid);
               }
             } else {
               @{$multipids{$uname}}=($instusers->{$uname}{'id'},$pid);
             }
```

```
                        } elsif ($pid =~ /^A\d{8}$/) {
                          $instusers->{$uname}{'id'} = $pid;
                        }
                      }
                    }
                  } else {
                    $instusers->{$uname} = {
                      firstname => $first,
                      lastname => $last,
                      id => $pid,
                      permanentemail => $uname.'@msu.edu',

                      inststatus => [$type],
                    };
                  }
                  if (defined($instids->{$pid})) {
                    if (ref($instids->{$pid}) eq 'ARRAY') {
                      if (!grep(/^$uname$/,@{$instids->{$pid}}))
              {
                        push(@{$instids->{$pid}},$uname);
                      }
                    } elsif ($instids->{$pid} ne $uname) {
                      @{$instids->{$pid}} = ($instids->{$pid},$uname);
                    }
                  } else {
                    $instids->{$pid} = $uname;
                  }
                }
                $outcome = 'ok';
            }
        }
           if ($ldapfilter ne '') {
              my $ldapres = &ldap_search($ldapfilter,$instusers,$types);
              if (!$dbflag) {
                 $outcome = $ldapres;
            }
        }
           return $outcome;
     }
```

At MSU, a search of the LDAP directory is used to supplement SQL queries of Faculty, Staff and Student database tables, because there are no student/employee IDs available from MSU's LDAP service. The LDAP search is used to retrieve information about users who

have MSUNetIDs (i.e., official usernames from MSU), but are not currently affiliated with any of the institutional user types, so are absent from the six SQL database tables.

```
sub ldap_search {
    my ($ldapfilter,$instusers,$types) = @_;
    my $outcome;
    my $ldap = Net::LDAP->new( 'ldap.msu.edu' );
    if ($ldap) {
        $ldap->bind;
        my $mesg = $ldap->search(
            base => "dc=msu, dc=edu",
            filter => $ldapfilter,
            attrs => ['sn','givenName','title','uid','mail','employeeType'],
     );
        if ($mesg->code) {
            $ldap->unbind;
            return;
     } else {
            $outcome = 'ok';
     }
        foreach my $entry ($mesg->entries) {
            my $uname = $entry->get_value('uid');
            next if ($uname eq '');
            my $first = $entry->get_value('givenName');
            my $last = $entry->get_value('sn');
            my $email = $entry->get_value('mail');
            my $type;
            if (($entry->get_value('employeeType') eq 'Faculty')
        || ($entry->get_value('employeeType') eq 'Staff'))
        {
              $type = $entry->get_value('employeeType');
        } elsif ($entry->get_value('title') eq 'Student')
        {
              $type = $entry->get_value('title');
        }
            if (ref($types) eq 'ARRAY') {
              if (@{$types} > 0) {
                if (($type ne '') && !(grep(/^$type$/,@{$types})))

                  next if (!grep(/^default$/,@{$types}));
                }
                next if (($type eq '') && (!grep(/^default$/,@{$types})));
              }
```

```
            }
               if (ref($instusers->{$uname}) eq 'HASH') {
                 if (ref($instusers->{$uname}{'inststatus'})
            eq 'ARRAY') {
                   if (!grep(/^$type$/,@{$instusers->{$uname}{'inststatus'}}))
            {
                     push(@{$instusers->{$uname}{'inststatus'}},$type);
                   }
                 }
               } else {
                 $instusers->{$uname} = {
                 firstname => $first,
                 lastname => $last,
                 id => '',
                 permanentemail => $email,
                 inststatus => [$type],
               };
         }
            $ldap->unbind;
      }
         return $outcome;
   }
```

## allusers_info

Three arguments are required:

1. $dom - domain
2. $instusers - reference to hash which will contain hashes, where keys will be usernames and value will be a hash of user information.

   Keys in the inner hash will be some or all of: lastname, firstname, middlename, generation, id, inststatus - institutional status (e.g., faculty,staff,student)

   Values are all scalars except inststatus, which is an array.
3. $instids - reference to hash which will contain ID numbers. keys will be unique IDs (student or faculty/staff ID).

   Values will be either: scalar (username) or an array if a single ID matches multiple usernames.

Returns 'ok' if no error occurred.

Side effects - populates the $instusers and $instids refs to hashes with information for all users from all available institutional datafeeds.

In the MSU case, six SQL database tables are queried via the *query_user_tables()* routine described above.

```
sub allusers_info {
    my ($dom,$instusers,$instids) = @_;
    my $outcome;
    my ($dbh,$dbflag) = &connect_DB('HR');
    if ($dbflag) {
        my @srchtables = ('FACULTY_VU','STAFF_VU','STUDENT','AFFILIATE',
 'ASSISTANT','STUDENT_AFFILIATE');
        &query_user_tables($dbflag,$dbh,\@srchtables,$instusers,$instids);
        $outcome = 'ok';
        &disconnect_DB($dbh);
 }
    return $outcome;
}
```

## 4.7 Search Filters for Official Course Categories

Courses in a domain can be self-cataloging if assigned an institutional code. For this to work two routines need to be customized in localenroll.pm - *instcode_format()* and *instcode_defaults()*. The first of these is used to split institutional course codes into their constituent parts, and populate some perl data structures with these data, which LON-CAPA can use to generate linked select boxes which users can use to create filters to apply when searching the "official" course catalog. The second routine constructs a regular expression used when searching for courses based on the filter chosen by the user, which will contain fragments of an institutional code.

### instcode_format

Six arguments are required:

1. domain ($dom)
2. reference to hash of institutional course IDs ($instcodes)
3. reference to hash of codes ($codes)
4. reference to array of titles ($codetitles),

   e.g., @{$codetitles}

   = ("year","semester","department","number")

5. reference to hash of abbreviations used in categories, ($cat_titles) e.g.,

   %{$$cat_titles{'Semester'}} = (

   fs => 'Fall',
   ss => 'Spring',
   us => 'Summer');

6. reference to hash of arrays specifying sort order used in category titles ($cat_order), e.g., @{$$cat_order{'Semester'}} = ('ss','us','fs');

The routine returns 'ok' if no errors occurred.

At MSU, "fs13nop590" is an example of an institutional course code; including the following entry in the instcodes hash passed in by reference - $$instcodes{'43551dedcd43febmsul1'} = 'fs13nop590' would cause the $codes perl data structure to be populated.

fs13nop590 would be split as follows:
$$codes{{'year'} = '2013'
$$codes{'semester'} = 'Fall'
$$codes{'department'} = 'nop'
$$codes{'number'} = '590'
The routine used at MSU is as follows:

```
sub instcode_format {
    my ($dom,$instcodes,$codes,$codetitles,$cat_titles,$cat_order)
  = @_;
    @{$codetitles} = ("Year","Semester","Department","Number");
    %{$$cat_titles{'Semester'}} = (
        fs => 'Fall',
        ss => 'Spring',
        us => 'Summer'
  );
    @{$$cat_order{'Semester'}} = ('ss','us','fs');
    foreach my $cid (keys %{$instcodes}) {
        if ($$instcodes{$cid} =~ m/^([suf]s)(\d{2})(\w{2,3})(\d{3,4}\w?)$/)
     {
            $$codes{$cid}{'Semester'} = $1;
            $$codes{$cid}{'Department'} = $3;
            $$codes{$cid}{'Number'} = $4;
            my $year = $2;
            my $numyear = $year;
            $numyear =~ s/^0//;
            $$codes{$cid}{'Year'} = $year;
            unless (defined($$cat_titles{'Year'}{$year}))
        {
              $$cat_titles{'Year'}{$year} = 2000 + $numyear;
        }
     }
  }
    my $outcome = 'ok';
    return $outcome;
}
```

### instcode_defaults

Three arguments are required:

1. domain ($dom)
2. reference to hash which will contain default regular expression matches for different components of an institutional course code

   ($defaults)
3. reference to array which will contain order of component parts used in institutional code. ($code_order)

The routine returns 'ok' if no errors occurred.

At MSU, the regular expression fragments used mirror those included in the regular expression used in instcode_format() to split an institutional course code into its component parts.

```
sub instcode_defaults {
    my ($dom,$defaults,$code_order) = @_;
    %{$defaults} = (
        'Year' => '\d{2}',
        'Semester' => '^[sfu]s',
        'Department' => '\w{2,3}',
        'Number' => '\d{3,4}\w?',
  );
    @{$code_order} = ('Semester','Year','Department','Number');
    return 'ok';
}
```

## 4.8 Validation of Requests for Official Courses

Course requests for official courses, i.e., courses with a valid institutional code, can be set to be processed automatically, on submission, if the requestor has been validated as the instructor of record for the course (based on institutional code).

To provide this functionality the following routines in /home/httpd/lib/pel/localenroll.pm will need to be customized.

*validate_instcode()*, *validate_crsreq()*, *crsreq_checks()*.

validate_instcode() is called when a request is being made for an official course. A check is made that the institutional code for which a course is being requested is valid according to the institutional schedule of official classes.

validate_crsreq() is used to check whether a course request should be processed automatically, or held in a queue pending administrative action at the institution.

Course requests will trigger this check if the process type has been set to "validate" for the course type (official, unofficial or community) and the requestor's affiliation. Whether "validate" is an available option in the Domain Configuration menu is controlled by crsreq_checks().

One scenario is where the request is for an official course, in which case a check could be made that the requestor is listed as instructor of record for the course in the institution's course schedule/database. This also involves validate_instcode(), but in this case the username of the course owner is also included, and a more restrictive test is used, namely that the requestor is listed as instructor of record for the course in the institution's course schedule/database.

Other scenarios are possible, and the routine can be customized according to whatever rules a domain wishes to implement to run validations against given the data passed in to the routine.

Customization of *possible_instcodes()* is also needed to support creation of dropdown lists used by the requestor when selecting the institutional code for the course to be created.

### validate_instcode

Two arguments are always required, and a third is needed if instructor of record status is being checked.

1. LON-CAPA domain that will contain the course
2. institutional code (in the MSU case this is a concatenation of semester code, department code, and course number, e.g., fs03nop590).
3. optional institutional username for the course owner.

An array is returned containing:

1. the result of the check for a valid instcode.
2. (optional) course description.

A valid instcode is confirmed by returning 'valid'. Otherwise a description of why the validation check failed can be returned for display to the course requestor. If no course description is available, " should be set as the value of the second item in the returned array.

### validate_crsreq

Six arguments are required:

1. domain ($dom)
2. username:domain for the course owner ($owner)
3. course type – official, unofficial or community ($crstype)
4. comma-separated list of owner's institutional groups ($inststatuslist)
5. institutional code ($instcode)
6. comma-separated list of requested institutional sections ($instseclist)

A valid courserequest is confirmed by returning 'process'. The following can be returned: process, rejected, pending, approval or error (with error condition - no :), followed by a : and then an optional message.

1. process - the requestor is the recorded instructor - create the course
2. rejected - the requestor should never be requesting this course, reject the request permanently
3. pending - the requestor is not the recorded instructor, but could become so after administrative action at the institution. Put the request in a queue and check localenroll:validate_instcode() periodically until the status changes to "valid".
4. approval - the request will be held pending review by a Domain Coordinator.
5. error (followed by the error condition).

If the response is pending then the course request is stored in a queue. If your domain is configured to process pending requests for official courses, once validated (see: 2.10 Auto-course creation settings), then the nightly run of Autocreate.pl will test each currently pending course request, to determine if the owner can now be validated, and if so, will create the course. If the owner remains unvalidated the request will remain in the queue. Domain Coordinators can display a list of requests for official courses, queued pending validation, via the "Course and community creation" page (see: 3.4 Creation Options).

### crsreq_checks

Three arguments are required:

1. domain for which validation options are needed. ($dom)
2. ref to array of course types – official, unofficial,community. ($reqtypes)
3. ref to a hash of a hash which will determine whether "validate" will be one of the possible choices for each course type - outer hash key, and institutional type - inner hash key ($validations).

   For example to allow validate to be a choice for official classes for Faculty, crsreq_checks would include:

   $validations{'official'}{'Faculty'} = 1;

   The institutional types are those defined for the domain in the domain configuration screen for: Default authentication, language, timezone, portal and types (see the 2.4 Defaults help page).

A value of 'ok' should be returned if no errors occurred. The routine used at MSU is as follows:

```
sub crsreq_checks {

    my ($dom,$reqtypes,$validations) = @_;
    if ((ref($reqtypes) eq 'ARRAY') && (ref($validations) eq 'HASH'))
  {
        my (%usertypes,@order);
        if (&inst_usertypes($dom,\%usertypes,\@order) eq 'ok') {
```

```
                foreach my $type (@{$reqtypes}) {
                   foreach my $inst_type (@order) {
                     if (($type eq 'official') && ($inst_type
                  eq 'Faculty')) {
                        $validations->{$type}{$inst_type} =
                    1;
                  } else {
                        $validations->{$type}$inst_type = '';
                  }
                }
            }
        }
    }
       return 'ok';

}
```

### possible_instcodes

The routine gathers acceptable values for institutional categories to use in the course creation request form for official courses.

Five arguments are required:

1. domain ($dom)
2. reference to array of titles ($codetitles), e.g.,

   @{$codetitles} = ('Year','Semester','Department','Number');

3. reference to hash of abbreviations used in categories ($cat_titles), e.g.,

   %{$$cat_titles{'Semester'}} = (
   fs => 'Fall',
   ss => 'Spring',
   us => 'Summer'
   );

4. reference to hash of arrays specifying sort order used in category titles ($cat_order), e.g.,

   @{$$cat_order{'Semester'}} = ('ss','us','fs');

5. reference to array which will contain order of component parts used in institutional code ($code_order), e.g.,

   @{$code_order} = ('Semester','Year','Department','Number');

A value of 'ok' should be returned if no errors occurred.