OpenCoral Introduction and Overview

Table of contents

1 OpenCoral

Welcome to OpenCoral. The OpenCoral Software System is a suite of software tools that is designed to help with the management and operation of advanced laboratories such as micro and nano fabrication facilities found in a number of universities.

2 What is OpenCoral?

For the user in the lab, it includes the following key capabilities:

- Allows equipment to be reserved in advance and to see who else has equipment reserved.
- Allows equipment to be enabled and disabled when in use to indicate to others that the equipment is in use.
- Report equipment problems and serious shutdown conditions.
- Quickly check on the operational status of each piece of equipment and examine more detailed reports of problem/shutdown conditions and their resolution.
- Allow checkout of laboratory supplies such as wafers, mask blanks, and storage containers.
- Optionally, collect and save run data during processing.

For the lab management and staff, it includes the following key capabilities:

- Maintain lists of qualified users on each piece of equipment.
- Allow certain users to have special privileges on specific pieces of equipment. For example:
  - An "operator" is allowed to charge others for equipment or staff time spent on their behalf.
  - An "instructor" is allowed to qualify other people to use that piece of equipment.
  - A "maintainer" is allowed to clear equipment problem and shutdown conditions.
  - An "engineer" is allowed to do all of the above.
- Generate detailed laboratory usage information including equipment reservations, equipment usage, staff and training activities, and equipment problems and shutdown condition for use in generating laboratory charges and helping to manage the facility.
- Specify which projects each lab member is allowed to work on and, in turn, specify which account each project is allowed to charge to.
- Subscription to monthly charged services such as storage locker facilities, monthly cardkey access charges, etc.
- Optionally interlock equipment so that it must be enabled in order to function properly. Of course, this hardware interlocking requires additional software and the appropriate equipment knowledge so that an appropriate interlock point can be selected.

- A reporting engine based on xReporter that supports online (html/xml), Excel, and PDF report formats. Additionally, flexible role-based report access is supported so that lab members only have access to data that they should see.

Because each facility is unique and likely has special needs, Coral is designed to be as flexible as possible and allows for many configurable capabilities. The range of features to support facility-specific flexibility include:

- Support for either Postgres or Oracle databases on either Linux or Solaris platforms.
- A flexible build-time configuration tool that allows most build- and run-time parameters to be specified.
- Support for a rich set of laboratory-wide and equipment-specific roles to control which privileges and capabilities each member may access.
- Flexible XML-based policies that capture the business logic of the Coral system without requiring any changes in the underlying Coral codebase. Examples of policies related to equipment reservations include:
  - Only qualified users can reserve equipment.
  - Non-staff members may only reserve equipment 10 days in advance
  - Users can only reserve a maximum of 10 hours on this popular machine.
  - Users can only reserve a maximum of 2 hours of time on a specific piece of equipment during "prime time" (e.g., 8 a.m. to 6 p.m. on Monday-Friday)
- Support for site-specific custom Java code for those sites wishing to incorporate features not currently supported in the core Coral capabilities.

3 Local and Remote Coral Clients

There are two versions of the OpenCoral client. Local Coral is a client/server version that can be run from within a sub-network and is the one that will normally be run within a particular facility. We will generally refer to it as either Coral or the Coral client in our documentation. Remote Coral is a version of OpenCoral client that can be run from anywhere on the Internet using a web browser that supports Java Web Start. We will call it Remote Coral in our documentation. Remote Coral is actually a fully functional Coral client that is downloaded onto the machine in question as a collection of jar files. Because Remote Coral is running on the local machine and has modest communication with the Coral servers, a Remote Coral client will perform well even with modest network bandwidth. Java Web Start handles monitoring and automatic download of new releases of the Remote Coral client. In general, all that is required to run Remote Coral on a particular machine is an appropriate Java Runtime Environment (JRE). Because Remote Coral is a Java application, it will run on virtually any platform including all Windows platforms, Macintosh OS X (and newer), all Linux machines, and on Solaris platforms (including sparc, x86, and x64 architectures).

What are the differences between the local and remote versions of the Coral client? In general, there are very minor differences between the local and remote versions of the Coral client. They each provide virtually identical functionality. The local Coral client receives direct notification of various events from each of the Coral servers. This insures that the local Coral clients are instantaneously notified of changes in equipment status, new reservations, etc. Because Remote Coral clients, are frequently run from behind a variety of firewalls, many Remote Coral clients will be unable to receive these event notification messages as most firewalls will view them as unsolicited incoming requests and will block the event notifications. Accordingly, Remote Coral asks the servers every few minutes for any updates in equipment and reservation status that have occurred. Accordingly, the Remote Coral client may be a few minutes out of date in terms of displaying some status information but, in practice, that has not been a complication.

The only other difference of significance between the local and Remote Coral clients is that, at present, a user can only set their Remote Coral password from a local Coral client. Because Java Web Start would allow anyone to download the Remote Coral client, this provides an additional measure of insurance that only valid lab users ... namely those who have actually set their Remote Coral password in your facility ... are able to run a Remote Coral client for your facility.

4 Overall Coral Architecture

Coral is a conventional three-tier (client/server/database) architecture that makes use of well-established standards for communications between client and servers and between servers and database. Specifically, Coral relies on the Object Management Group's (OMG) Common Object Request Broker Architector (CORBA) standard for communications between client and servers and server to server. For communications between servers and database we rely on the Java Database Connectivity (JDBC) standard communications.

Although they are invisible to the client, at present there is a separate Coral server for each range of functionality. Moreover, each server has a corresponding database user that owns the database tables that are critical to it's area of inflence. That database user is typically the only entity with write-access to those tables. In limited cases, other database users are given read-only access to a particular server/database user's tables.

At present, the set of servers (with the corresponding database user in parenthese) that constitute the Coral system include:

- Administration manager (admmgr)
  - The admin manager is the first server that starts and controls and monitors all of the other servers.
  - Each client and all of the other servers first registers with the admin manager and periodically sends a "heartbeat" signal to it.

    - If a server fails to send a hearteat in an appropriate time period, the admin manager will attempt to restart all of the other servers.
- Authentication manager (athmgr)
    - The auth manager handles authentication of Remote Coral passwords and the encrypting/decrypting of those passwords.
    - The auth manager also handles the optional NIS authentication of users if the NIS-authentication open is used for local users. Note: by default, Coral relies on normal OS authentication of users for local Coral authentication and, as a result, actual password authentication is not handled by the auth manager in these cases.
    - Once a lab member authenticates themselves to the authentication manager, the auth manager generates a "ticket" that lists the set of roles, privileges, projects, and accounts that are appropriate for that member. The ticket is an XML document. Giving each Coral client a ticket for that lab member greatly reduces the number of times taht a member will have to make a call to a server and database.
- Equipment manager (eqmgr)
    - The equipment manager handles the requests to enable/disable each piece of equipment.
    - The equiment manager also handles tracking the state of functionality of each piece of equipment. Coral uses a very simple model of equipment status:
        - *Green light:* The equipment is up and functioning normally.
        - *Yellow light:* The equipment has one or more outstanding problems that deserve attention but that are not so severe so as to keep the equipment from being used judiciously. For example, exposure uniformity on a mask aligner may not be within spec, but, for someone doing a non-critical lithography, this may not be problematic.
        - *Red light:* The equipment suffers from a serious failure that prevents the equipment from running normally and requires a repair before anyone can use the system.
        - *Orange light in the red position:* A facility required by the piece of equipment is down. Note: a facility is also treated as a piece of equipment even though it is not normally enabled/disabled in Coral. Typically, facilities might include things like house nitrogen, deionized water, chilled water, or fume scrubbers. A piece of equipment in this condition cannot normally be enabled.
        - *Orange light in the yellow-light position:* An optional facility used by the piece of equipment is down. A piece of equipment in this condition can normally be enabled as the facility in question is not required. Coral trusts that the lab member will be astute enough in these cases to deteermine whether the problematic facility will have an impact on their use of the equipment and will act accordingly.
    - The equipment manager also controls the equipment roles that are assigned to each lab member for each piece of equipment.

- Event manager (evtmgr)
  - The event manager is the central nervous system of Coral and notifies all other Coral servers and local Coral clients of events that may be of interest to them.
  - Because of potential firewall problems, Remote Coral clients will periodically poll the event manager for recent events of interest to them.
- Hardware manager (hwrmgr)
  - The hardware manager is an optional server that will only run if one or more pieces of equipment is using harware interlocks to control enables and disables of equipment.
  - The hardware manager will send eneable/disable requests to the *hardware daemon* and then wait for the appropriate success/failure returns from the hardware daemon.
- Policy manager (polmgr)
  - The policy manager loads an XML policy file for each of the key servers when it starts up. Each of the other managers will send *policy requests* to the policy server to inquire as to whether a certain action is permitted or denied.
  - The policy files encapsulate the business logic for each facility in a way that the behavior or Coral may change dramatically from site to site without requiring any changes to the underlying Java codebase.
  - The policy manager is based upon the XML Axess Control Markup Language (XACML) standard written by OASIS.
- Reservation manager (resmgr)
  - The reservation manager controls and tracks requests for equipment reservations (subject, of course, to the subsequent approval of the policy manager.
- Resource manager (rscmgr)
  - The resource manager is used to create and deactivate members, projects, and accounts.
  - The resource manager also tracks the relationships between members and projects (that is, which members are allowed to work on which projects) and, in turn, the relationships between projects and accouts (that is, which projects are allowed to charge to which accounts).
  - The resource manager also controls the assignment of non-equipment roles to varous members.
- Runtime manager (runmgr)
  - The runtime manager is an optional server that controls the collection of process data that may be required each time a piece of equipment is used. For example, it may be desirable to collect the number of wafers run in a piece of equipment or the thickness of deposited material in a sputtering machine.
  - The information that is collected is specified by an XML document that contains a separate specification of each piece of equipment for which data is to be collected. We currently support text fields, integers, floating point numbers, pulldown menus, and checkboxes.

- Service manager (svcmgr)
  - The service manager supports a rudimentary capability to check in and check out items for the purposes of inventory control.
  - These items may be items that are ultimately recharged to the members (wafers, for example) or they may be items that are required in the facility but may not be explicitly recharged to the users (sulfuric acid, for example)
  - The service manager also handles subcriptions for various services. In general, a service can be something for which a fixed monthly charge will be generated. For example, some facilities may charge a modest monthly charge for maintaining a key or access card.
- Staff manager (stfmgr)
  - The staff manager tracks time spent by staff members working on behalf of others. That time is often ultimately charged to that lab member.
  - The staff manager differentiates between "normal" staff time and time spent training other people because there may be a different charge rate for training than their is for conventional process support. Additionally, it is often useful to be able to discern the fraction of time staff members spend training other users to operate equipment as opposed to the time that they are actually operating equipment on behalf of others.

5 Server and Database Considerations

We have worked hard to make Coral a flexible system in a variety of different facilities. That flexibility extends to choices of hardware for the servers, database choices and approaches to client deployment. This section will describe a few of those choices.

In general, we assume that the Coral database and all of the Coral servers will run on a single machine. Architecturally, that is not at all required, but, in most cases, a single machine is more than adequate to run all Coral servers and the database. This machine can be either an x86 machine running a Linux operating system or a Sun machine (either SPARC or x86 platform) running a Solaris operating system. Furthermore, we support either Postgres or Oracle databases on either platform. At present, most of the sites currently running Coral are running on a SPARC/Solaris platform running the Oracle database. Because we believe that the most cost effective platform is a x86 Linux maching running a Postgres database, the documents on this site provide instructions for installing and configuring Coral on a Linux platform running a Postgres database. In particular, we assume that folks will be running their Coral servers and database on a machine running RHEL5 (RedHat Enterprise Linux) with the version of Postgres currently supported by their `pup` (package updater) utility. We believe, however, that these instructions are largely correct for just about any 2.6 kernel platform. Choices of Linux version other than RHEL5 should be made only if you are

sufficiently confident in your ability to install, configure and run Coral without further input from us.

> **Note:**
> Academic licensing of RHEL5 is available from RedHat at an annual subscription price of $60. As far as Coral is concerned recent versions of RHEL5, Ubuntu, and CentOS have all been found to be suitable.

Solaris 10 is an equally attractive operating system for Coral and, in fact, most sites currently running Coral in production do so on Solaris/SPARC servers. Recently, Sun has announced that they are fully supporting Postgresql on Solaris 10 servers which makes Solaris 10 a perfectly viable OS for Coral in our view.

In terms of machine performance, just about any modern hardware with one or more processors should be adequate. We do suggest that the Coral database/server machine should have a minimum of 2GB of main memory to avoid unnecessary swapping. Of course, items that affect reliability and uptime including RAID disks, hot-swappable disks, redundant and/or hot-swappable power suppplies are also desirable.

> **Note:**
> Disk and database backup and other elements of disaster recovery are essential to any successful computer system. We do not claim to be experts in this arena but we do counsel that these matters be taken very seriously. A serious disk crash or other serious hardware failure could both jeopardize your daily operations for many days if you do not have adequate redundancy or backup measures in place. Moreover, without serious and comprehensive data backup, you could risk losing years of historical records related to the usage and charges associated with your facility. As most universities are subject to potential audits years after the fact, serious attention should be given to appropriate hardware and data backup strategies.

We recommend that the Coral database and servers be run on a machine that is *not* accessible to a large community of users both so that extraneous activities not compomise the performance of this system and so that there are fewer security challenges associated with this important machine

> **Note:**
> In general, this machine cannot be competely hidden behind a firewall unless all potential users (including Remote Coral users) will be accessing that machine from behind the same firewall. That said, even though Coral clients may need public access to the Coral servers, the JDBC database connections (which, in general, are made between servers and the database) may be on a private network to minimize the chances of unwarranted access to the Coral database.

6 Coral Client Deployment Considerations

We expect that most facilities will wish to deploy local Coral on a number of machines in their facility. There is no requirement that there be one Coral client per piece of equipment in the facility but there likely is the requirement that there be one Coral client per person in the facility under most conditions. Because people in the laboratory move from tool to tool over the course of the day, it is important to consider how to accommodate that mobility in a facility.

For many facilities, deploying a set of x86 platforms running a Windows OS may be the natural choice. While Coral runs perfectly well on this platforms, this choice may suffer from a couple of complications:

- Windows machines are not the best in terms of supporting multiple users. In particular, setting up a large number of users on multiple Windows machines sounds challenging at best. Using a centralized NIS authentication mechanism may be an attractive alternative in these cases.
- Windows machines are often more problematic in terms of both control of spam and viruses as well as in terms of controlling the installed software base on each machine.
- Windows platforms do not offer a good strategy that supports mobility in the laboratory and easy migration of a Coral client from location to location in the facility.

We have found the Sun Ray thin clients from Sun to be a very attractive solution in the laboratory setting. In particular, their support of "Hot Desking" by allowing an individual's session to move with them from station to station over the course of a day is a powerful asset in the laboratory. The support of the smartcards used for this hot desking requires zero administration (they are, in fact, shared, common-use smartcards). Moreover, all of the Sun Rays are supported by a single server so overall system administration load is reduced. Finally, if one Sun Ray fails, a replacement can be plugged in with zero administration. There are no configurable elements on an individual Sun Ray ... not even an IP address.

The list price of a Sun Ray 2 is currently $249 plus the cost of a VGA monitor or LCD display. Universities will typically expect a significant price reduction. Adopting the Sun Ray approach will require an additional server that becomes the Sun Ray server. This machine will have login accounts for all users of the facility and will support not only their Coral clients but their browsing and desktop applications. We have found it desirable to have at least 4GB of main memory on this machine in order to comfortably support 25-30 Sun Ray clients. For both security and performance reasons, we believe that it is important that the machine that supports the Sun Ray clients *not* be the same server that hosts the Coral servers and database. Originally, Sun Ray Server Software ran only on Solaris/SPARC platforms. However, according to the latest infomration from Sun, it will now also run on Solaris/x86 and on Linux/x86 platforms. For desktop applications, Star Office is fully compatible both in

look and feel and in file-compatibility with familiar Windows desktop applications including Word, Excel, and PowerPoint. Finally, although we have not tested it, Sun now supports deployment of full Windows services onto the Sunray by using Windows Terminal Services and the Sunr Ray Connector for Windows.

7 Coral Report Generation

Coral users and Coral laboratory administrators will want to extract a variety of reports from their Coral installation. Because we believe that Coral's reporting needs are not fundamentally different from those of most other systems running on top of a relational database, we have elected to use an open source reporting engine rather than develop a Coral-specific reporting capability. In particular, the Coral reporting system is based on the xReporter reporting engine. Information about xReporter is available at http://xreporter.cocoondev.org. Xreporter supports a variety of nice features including:

- It is a true web application based on top of Apache's Tomcat servelets and Cocoon display technology.
- It generates output online (HTML/XML) in the form of PDF documents or true Excel files (not simply tab-separated data files).
- It uses your Remote Coral password for authentication and has full support of Coral roles to control which reports are visible to which people in your facility. With proper setting of roles, you don't have to worry about people seeing more information than they should.
- Reports are described as XML documents making it easy to upgrade or enhance existing reports and to generate new ones.
- The default Coral xReporter installation uses SSL-encryption so that all passwords and report information are transmitted securely over the network.
- Because it uses JDBC to connect to the database, the Coral xReporter installation may be hosted either on the same machine that is used to run the Coral database and servers. Our default installation assumes that this is a Linux platform, but xReporter has been successfully installed on Solaris platforms as well.

The Coral xReporter reporting engine is available as a separate CVS module named `coral-xr` available from the opencoral.mit.edu CVS server.

8 What you need to know to install and run Coral

Coral is a sophisticated and flexible software tool that runs on multiple platforms against multiple databases. In order to install, manage, and maintain it effectively you (or someone in your organization) needs a certain level of skill to be an effective Coral administrator. While we have tried to provide detailed documentation about installing and configuring Coral and

related software tools, we assume a certain level of familiarity with a variety of software tools.

In particular, it is useful if you have the following skills:

- Familiarity with Linux or Solaris, as appropriate. We assume that you know enough about the underlying operating system to comfortably move around the file system, understand permissions, know how to start/stop services, know how to edit files using a simple editor such as vi, vim, or emacs, know how to keep your operating system appropriately patched, etc. If you plan to install Coral on an operating system other than RHEL5, we trust that you know enough to adapt the RHEL5-specific instructions included here to your own operating system.
- Familiarity with a SQL database. Coral supports either Postgresql or Oracle databases. We assume that you have some familariaty with database administration, can read and understand basic SQL commands, etc. Most people running Coral choose to do so on an Postgresql database. At the moment, we have folks running Coral on Postgresql databases as old as version 7.4 and as new as version 8.4. There have been a number of changes in SQL syntax and database configuration over that time and it is difficult to keep our documentation up to date with all of those changes. While we try to make updates when we are aware of changes, it is incumbent on you to be on the lookout for propbems of this type, particulaarly if you are running a version of Postgresql that is either significantly older or newer than the standard version that we support.
- Familiarity with XML files and schemas. The most powerful and flexible customization features in Coral are all controlled by XML documents. This includes the customization of policies for making equipment reservations, policies for how you calculate charges for lab usage based on equipment and other resource usage, and any data that you may wish to collect during equipment usage. While you will often be able to cut-and-paste from XML files that other sites running Coral have customized, a basic understanding and familiarity with XML and XML schemas is useful. It is also useful to have access to an XML-aware editor such as oXygen (http://www.oxygenxml.com).