Sakai 11 System Requirements

Sakai 11 Browser Support

Sakai 11 is designed to work with modern browsers. Recommended browsers (those with the most QA testing effort against them) are Google Chrome and Mozilla Firefox. Internet Explorer 11 (IE 11) and later Microsoft browser options should work fine. If you are upgrading from an earlier version of Sakai (pre-Sakai 11) please make sure that you don't have the property X-UA-Compatible:IE set, or it may make Microsoft IE 11 behave as though it is an earlier browser version and therefore not work.

Some iPad and mobile testing has been done, but more work is needed in this area (this is the first release of Sakai using responsive design).

Operating system (OS) choices

Sakai is OS neutral. It is typically run on any of the numerous Linux distributions. Examples of common Linux distributions include as CentOS, Debian GNU/Linux, Fedora, Gentoo Linux, Red Hat Enterprise Linux (RHEL), SuSe Linux, Ubuntu. Sakai can also run on Mac OS X server, Microsoft Windows and Sun Solaris. Operating systems other than Linux are not nearly as well tested, and all of the community QA servers are running Linux, so this is generally the recommendation.

Java 8

Sakai 11 has been tested most thoroughly with Oracle's Java 8. Requires JDK 8 to build "all code". Certain files, such as *.jsp and *.jws, require compilation so downloading and attempting to use only the run time environment (e.g. JRE 7.0) will not suffice. Sakai 11 was QA tested with Java 1.8.0_77 .

Application server choices

Apache Tomcat 8 is the recommended, and most thoroughly tested servlet container, and is most often used with a web server like Apache Http Server. Several schools run their Tomcats with nginx proxies without issue. Sakai 11 was QA tested with Tomcat 8.0.32 .

Database choices

Sakai production installations typically run MySql 5.6 or later. Oracle is the next most popular choice, Oracle 12c.

Database support details

Clustering, file storage and load balancing strategies

A typical Sakai cluster is comprised of one or more application servers running one or more instances of Tomcat 8 operating either in standalone mode or behind the Apache HTTP 2.4 web server. Each Tomcat instance runs a full copy of Sakai. The cluster is backed by a single database providing a transactional store of information.

Storing binary content outside the database is a configurable option in Sakai and highly recommended from a performance standpoint. Most Sakai schools with sizable user populations opt for network-attached storage (NAS) or storage area network (SAN) solutions. Load balancing is provided by Apache (using mod_jk, mod_proxy_balancer or mod_proxy_ajp) or dedicated hardware solutions such as F5 BIG-IP, NetScaler or Zeus.

External Authentication choices

Sakai can integrate with a variety of external authentication services including CAS, Kerberos, LDAP, Shibboleth and WebAuth.

Integrating with student information systems

Sakai Community institutions have integrated their Sakai installations with Banner, Datatel and Peoplesoft as well as a variety of home-grown student information systems (SIS).

Sakai has two basic approaches to integrating data from external systems. Most sites use a combination of these approaches. The first approach is to use the internal Sakai "provider" APIs. These APIs are places for Sakai to "consult" while Sakai is running. There are APIs for User Identity, User Directory, Course Listing and User Roles.

User Identity API: allows Sakai to call local code to validate users when they log into the system. This commonly uses Kerberos, Active Directory or LDAP to validate the user's credentials.

User Directory API: allows user information such as name and e-Mail address to be retrieved from an external system such as LDAP or X.509. The User Directory API has provisions to allow the local site to make decisions when to display student information in order to meet FERPA requirements. Each institution has different interpretation of FERPA so the precise FERPA decisions are delegated to the User Directory API.

Course Listing API: consulted when the instructor is creating a course site - this API returns the list of externally stored rosters for which the current user is the instructor. The user can select from one or more of these external rosters to associate with the course they are creating.

User Role API: is consulted when users log in to determine which external rosters they user is a member of and what their role is within those rosters. The Sakai internal configuration is updated if there are any changes to an individual's roster status.

The above API's are "pull" APIs--they are consulted when the user logs in or tries to take some action. The Course List API described above does not auto-populate courses.

If there is a desire to "push" information into Sakai, there are two approaches - Quartz and web services.

Sakai utilizes an internal batch system called Quartz that provides a cron-like capability within Sakai. Quartz is used by creating a Java class that does the necessary work and then having Quartz schedule the regular execution of that Java code.

A more common approach to pushing data (e.g. enrollment and course information) into Sakai is through web services. Many of Sakai's APIs can be accessed by web services. Web service access points have been developed for many of the common Sakai APIs used for configuration. These REST and SOAP web services can be called from PHP, Python, Perl, Java, .NET or any other language. The Sakai web service data structures are kept simple to insure the widest possible interoperability with as many languages as possible. Administrators often build scripts to pull data from their SIS system and populate Sakai with that data. These scripts may be automated using cron or manually executed by the administrator at the proper time during a semester.

This combination of pull/push configuration capabilities allows for a very wide range of integration possibilities for Sakai.