Blackboardlearn+

Release 9.1

Authentication for SP7 and Earlier

Blackboard

Publication Date: March 27, 2012 Revision: 0

Worldwide Headquarters	International Headquarters
Blackboard Inc.	**Blackboard International B.V.**
650 Massachusetts Avenue NW Sixth Floor Washington, DC 20001-3796	Paleisstraat 1-5 1012RB Amsterdam The Netherlands
+1 800 424 9299 toll free US & Canada	
+1 202 463 4860 telephone	+31 (0) 20 788 2450 (NL) telephone
+1 202 463 4863 facsimile	+31 (0) 20 788 2451 (NL) facsimile
www.blackboard.com	www.blackboard.com

Contents

Introduction

The Blackboard Learn authentication framework is provided using Building Block technology with full user interface installation, management, and Logging. This use of Building Blocks to provide authentication integration removes barriers and issues with system management related to custom authentication.

Blackboard recommends creating new authentication providers using the Building Block authentication framework and rewriting any custom authentication modules to utilize the new framework. If this is not viable, you can use the legacy framework to manage authentication.

In This Guide

This guide is divided into two parts: The first part explains the Blackboard Learn Authentication framework, the second part explains using the Authentication providers supported.

Important Note: This guide will not be updated. The Authentication Framework was updated as the release of Blackboard Learn Service Pack 8. Please see Authentication in help.blackboard.com for information on the new Authentication Framework.

About The Authentication Framework

The Blackboard Learn authentication framework is provided using Building Block technology with full user interface installation, management, and Logging. This use of Building Blocks to provide authentication integration removes barriers and issues with system management related to custom authentication.

The Blackboard Learn authentication provides support for the following authentication providers:

- LDAP (default)
- The internal Learn challenge-response (default)
- CAS
- Shibboleth
- Windows Auth
- Webserver Delegation
- Datatel
- OpenID

The authentication framework enables users providing ID and password credentials to validate and initiate a session in Learn, as well as the enterprise-level feature of integrating Blackboard Learn with one or more external authentication providers. The administrator arranges the providers in order of preference enabling an authentication cascade where each provider is sequentially queried until the user is logged in or fails to be authenticated.

Configuration and Management

Authentication configuration and management is accomplished through a consistent and familiar user interface. The administrator manages authentication Building Blocks through the Building Block manager, and configures and manages installed authentication providers via the Authentication management user interface. An administrator is able to add, remove and configure one or more authentication providers without requiring access to the filesystem or services to be restarted. This interface provides information about the current state of each installed provider, allows configuration values to be managed for each provider, allows providers to be chained together in a particular order, and facilitates the integration being put into active or inactive states. This interface also allows access to provider level logging of authentication activity.

CAS Support

Central Authentication Service (CAS) is the most common centralized web authentication Single Sign On (SSO) protocol for intra-organization authentication. CAS is configured by an administrator as an authentication provider in the authentication management interface.

SunGardHE Luminis 5 supports CAS, simplifying Luminis to Learn SSO.

Shibboleth Support

Shibboleth is a standards-based, open source software package for web single sign-on across or within organizational boundaries. An administrator can install their Blackboard Learn application to support Shibboleth and configure Blackboard Learn to use the Shibboleth provided credentials. The client configured Shibbileth installation, in conjunction with client additions to Apache 2 or IIS, enable the passing of login information to Blackboard Learn.

Shibboleth support in a Linux environment requires the installation and configuration of Apache 2. Blackboard is providing a supported documentation approach for administrators to install and configure Apache 2 to work with Learn 9.1 SP8 (see below section Apache 2 Support). This configuration in combination with the installation of the Shibboleth authentication provider enables the use of Shibboleth as a SSO provider for Blackboard Learn.

Emergency One-time Login

There may be occasions where administrators require login access to a system for which their original account is no longer available to them. The Emergency One-time login URL tool enables this access by providing a way for the administrator to create a one-time, session limited, login for a user from the command line. This ensures that administrators may have continued access to their system in the event of lost passwords and or failed authentication providers.

Use of this tool is specifically logged in the **Authentication Framework Logs**.

Logging

An administrator has access to logged information on authentication attempts for diagnostic and forensic purposes. The information includes details about logins and login attempts, logouts and session expirations, as brokered by an installed authentication provider.

Users can be given access to the authentication logging interface from the **Administrator Panel** under **Logs**, without being given access to manage (create, delete) authentication providers.

Authentication events logged to file are standardized for consumption by external log parsing tools.

APIs for custom authentication development

Should an organization require an authentication using a provider for which a solution is not delivered by Blackboard Learn, they may utilize the authentication APIs to develop a solution that may be installed, configured, and managed similar to those provided by Blackboard Learn.

Viewing Authentication Configuration

View the authentication configuration of Blackboard Learn from this page.

How to View Authentication Configuration

On the Administrator Panel, under **Building Blocks**, click **Authentication**. All authentication types are listed. The green check mark identifies the currently enabled authentication type. While it is possible to view settings for all the different Authentication Types, only one type may be enabled at a time. The default Authentication Type is Blackboard Challenge-Response.

Planning Authentication Implementation

This topic includes information on how the different authentication types (Blackboard default authentication, LDAP, and Web server delegation) are implemented. This information may be helpful to developers creating a custom authentication for Blackboard Learn.

Blackboard Default Implementation

The default implementation (realized in the class `BaseAuthenticationModule`) is implemented as a challenge response protocol, using a form submitted by HTTP-POST. This is a mechanism that avoids sending the actual password over the network in an unprotected fashion. In a naïve authentication implementation, username/password combinations would simply be transmitted across the network in clear text. The problem with this method is that malicious users would be able to see the username and password.

To facilitate greater security the authentication framework generates a pseudo-random number for each authentication attempt that is MD5-hashed against the servlet engine's session ID. This "challenge" string is sent to the client. The challenge string increases system security by improving the chances that the transmitted code has not been tampered with. When the user enters their password, it is MD5-hashed, then that hash string is combined with the challenge string sent by the server, and the resulting string is MD5-hashed. The resulting string is cryptographically secure in that the hash is one-way (MD5). This means that it is not possible to reconstruct its inputs, or to find inputs that result in the same hashed value. This compound hash is called the "response."

The server, when it receives the client response, performs the same calculations the client performed (except on the server-side, the password is already hashed). Additionally, the challenge string is also re-calculated from the stored pseudo-random number (to help prevent session hijacking). If the results match the client response, the authentication is successful.

LDAP Implementation

Blackboard has implemented a simple LDAP authentication module that uses data from the `authentication.properties` configuration file to bind to an LDAP server, or series of LDAP servers, and perform a lookup of a given user. To learn more about the configuration settings for LDAP authentication, see Setting Up LDAP Authentication Properties.

Web Server Delegation Implementation

Blackboard has implemented a Windows-specific Web server delegation authentication module that assumes the installation of a Blackboard-created ISAPI filter on the Web

server; the ISAPI filter populates a request header containing the Active Directory username for the Windows user. The `WindowsAuthModule` authentication module parses the request, extracts the unique identifier and attempts to match that identifier to a user in Blackboard Learn database.

`WindowsAuthModule`'s parent class, `ExternalAuthModule`, is a simple, general implementation for Web server-delegated authentication. It assumes that a module or filter has been installed on the Web server that populates the Computer Gateway Interface (CGI) variable REMOTE_USER in the request headers.

Extending Other Blackboard-created Authentication Modules

Extending other Blackboard-created implementations such as `WindowsAuthModule` is not permitted at this time.

Sample Custom Authentication Module

The following is an example of a Custom Authentication Module:

```
package blackboard.authentication.test;


import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;


import blackboard.platform.BbServiceManager;
import blackboard.platform.RuntimeBbServiceException;
import blackboard.platform.config.ConfigurationService;
import blackboard.platform.log.LogService;
import blackboard.platform.security.authentication.BaseAuthenticationModule;
import
blackboard.platform.security.authentication.BbAuthenticationFailedException;
import
blackboard.platform.security.authentication.BbCredentialsNotFoundException;
import blackboard.platform.security.authentication.BbSecurityException;
import blackboard.platform.security.authentication.HttpAuthConfig;
import blackboard.platform.security.authentication.SessionStub;


    /**
     * @author Blackboard Development
     *
```

```
     * A Sample Custom Authentication Module.
     */
public class CustomAuthModule extends BaseAuthenticationModule {

    //Save the log service so that we can log events.
       private LogService _logger;

    /*
     * Module initialization. This method gets called when Tomcat starts up.
     */
       public void init(ConfigurationService arg0) throws
        IllegalStateException {
        //Although this is not necessary for subclasses of
        //BaseAuthenticationModule, it is a good practice to do this
        //unless there is a reason not to.
        super.init(arg0);
        try {
        // Set up logging
          _logger = BbServiceManager.getLogService();
        } catch (RuntimeBbServiceException e) {
          e.printStackTrace();
         }
        if (_logger == null) {
           //This println statement will output to the
          //stdout-stderr.log file.
          System.out.println("logger is null");
        } else {
          _logger.logWarning("Custom Auth Module: init()");
        }
    }

    /*
     * Does the work of authenticating the user.
     */
       public String doAuthenticate(
              HttpServletRequest request,
```

```
                HttpServletResponse response)
                throws BbSecurityException,
                  BbAuthenticationFailedException,
                  BbCredentialsNotFoundException {
          //Since this module uses the standard Learn login form,
         //use the superclass implementation. The authenticated() method is
then
         //overridden to customize behavior. Other authentication modules
         //could do something else here if needed, as long as the
         //userid is returned or an exception is thrown.
         _logger.logWarning("Custom Auth Module: doAuthenticate()");
                return super.doAuthenticate(request, response);
       }

         /*
          * Gets called when the user explicitly logs out.
          */
          public void doLogout(
         HttpServletRequest request,
         HttpServletResponse response)
          throws BbSecurityException {

          _logger.logWarning("Custom Auth Module logging out");
         //perform custom logout work here
       }

          /*
          * Gets called when the user needs authenticating.
          */
       public void requestAuthenticate(
                HttpServletRequest request,
                HttpServletResponse response)
         throws BbSecurityException {
         //Do the superclass implementation, redirect to the Default Login
         //Page
         _logger.logWarning("Custom Auth Module: requestAuthenticate()");
         super.requestAuthenticate(request, response);
```

```
        }

          /*
          * Returns a String containing the authentication type.
           */
        public String getAuthType() {
               return "custom";
        }



         /*
         * Returns a String array of properties used by this authentication
         *  type.
          */
        public String[] getPropKeys() {
         String[] props = {"impl", "prop1"};
          return props;
        }

          /*
          * Blackboard Learn calls this to hand the authentication module the
          * configuration properties. This gets called before init so that it
can
          * use its properties if needed.
           */
        public void setConfig(HttpAuthConfig config) {
         //Just use BaseAuthenticationModule implementation. It will set
         //the member variable _config. Most modules that extend
         //BaseAuthenticationModule need not override this method.
         super.setConfig(config);
        }

          /*
          * Overrides BaseAuthenticationModule to do something different.
           */
        protected String authenticate(
```

```
      String username,
              String password,
              SessionStub sessionStub,
              boolean useChallenge)
              throws BbAuthenticationFailedException,
              BbSecurityException {
      //Do authentication logic here; validate password against an external
      //source... Could also call the superclass implementation to "fall
      //back" to Blackboard.

      //This implementation just tries out the exceptions that can be
      //thrown.
      if (username.equals("error1")) {
        throw
        new BbAuthenticationFailedException("Custom auth module, error1.");
    }

      if (username.equals("error2")) {
          throw new BbSecurityException("Custom auth module, error2.");
      }

      if (username.equals("error3")) {
        //null parameters, should display standard message.
        throw new BbAuthenticationFailedException();
    }

      //otherwise, return the user passed without checking. The user must
     //exist in Blackboard Learn for this to work properly.
    return username;
       }
} //End CustomAuthModule
```

Sample IUserPassAuthModule Code

You need to implement `IUserPassAuthModule` in a custom authentication module and add the `getUserFromUsernamePassword(x,y)` method as follows:

```
public class CustomAuthModule extends BaseAuthenticationModule
```

```
implements IUserPassAuthModule{
public User getUserFromUsernamePassword(String username, String
password)
throws PersistenceException, BbAuthenticationFailedException,
BbSecurityException {
// Required when implementing IUserPassAuthModule
// IUserPassAuthModule is necessary to ensure
compatibility with Content System (WebDAV)
String validatedUsername = authenticate(username,
Base64Codec.encode(password), null, false);
if (validatedUsername == null)
return null;
User user;
try { user =
UserDbLoader.Default.getInstance().loadByUserName(validatedUsername); } catch
(KeyNotFoundException e) { return null; } catch (Exception e) { return
null; } }
return user;
}
}
```

Default Authentication

Using Blackboard Learn Authentication

The default authentication for Blackboard Learn authenticates the user's login credentials against the Blackboard Learn database.

Customize the Default Authentication

Changing the Blackboard Learn Authentication process and options does not require any database changes. All of the options are stored in a properties file. Modify the `authentication.properties` file to customize the default authentication for Blackboard Learn.

Return to the Default Authentication

If, in the Course of setting up a customized Authentication solution, it is necessary to return to Blackboard Learn default authentication (`rdbms`), the authentication type (`bbconfig.auth.type`) can be set via the command line. This allows Blackboard Learn, at start up, to select the appropriate set of `auth.type*.*` entries. Follow these steps to reset the system to use the default authentication model:

1. Change to the following directory:

   ```
   cd BB_DEPLOY_DIR\blackboard\config
   ```

2. Edit the `authentication.properties` file as follows:

   ```
   auth.type.rdbms.impl=blackboard.platform.security.authentication.BaseAuthenticationModule

   auth.type.rdbms.use_challenge=true
   ```

3. Edit the `bb-config.properties` file. Change the `bbconfig.auth.type` property as follows:

   ```
   bbconfig.auth.type=rdbms
   ```

4. Run `PushConfigUpdates` to activate the changes. To learn more, see Appendix A: PushConfigUpdates.

Authentication Properties

The following table describes the available properties for the default authentication model. These properties are configured through the `authentication.properties` file. The `authentication.properties` properties file is found in `blackboard_home/bbservices/config`.

Property	Description
auth.type.rdbms.impl	Defines the class which must conform to the HttpAuthModule interface. The default value, blackboard.platform.security.authentication.BaseAuthenticationModule, should not be changed unless the Institution builds and implements its own class for Blackboard Learn default authentication.
auth.type.rdbms.use_challenge	Defines the encryption setting where a value of false indicates the password is encrypted with base 64 and a value of true indicates the password is encrypted with MD5. The default value is true. MD5 encryption offers stronger security for passwords. Base64 is similar to sending the password in plain text.

About Authentication Object Models

There are three objects used in Blackboard's default authentication type:

- HttpAuthManager
- HttpAuthModule
- BaseAuthenticationModule

The HttpAuthManager class and the HttpAuthModule interface cannot be modified. Institutions can plug-in their own module for a supported authentication type.

Authentication Object Model

The BaseAuthenticationModule class implements the HttpAuthModule interface and supports the default authentication type. The following object model depicts the HttpAuthModule implementations for LDAP and Web server delegation. IIS Web server delegation support is implemented in the class WindowsAuthModule. Apache Web server delegation support is implemented in the class ExternalAuthModule.

Note: Although WindowsAuthModule are public Java classes, it is not permitted to extend these classes. Most authentication modules will extend BaseAuthenticationModule. The PassportAuthModule is not supported in Release 7.0 and higher.

LDAP

About the LDAP Authentication Module

Standard LDAP (Lightweight Directory Access Protocol) authentication is fully integrated with Blackboard Learn. All necessary .jar files, including those for setting up an SSL connection between Blackboard Learn application server and the directory servers, are provided in the `/systemlib` directory. The application server startup executables include the .jar files in their classpath. Note that all configuration options in the `authentication.properties` file are set to default values. Some of these default values are place holders and must be changed by the Administrator for LDAP authentication to work successfully.

To begin authenticating against an LDAP server or servers, set the properties found in the `authentication.properties` file. The SSL Configuration topic has specific information on enabling the Blackboard Learn application server and the directory servers to communicate over SSL.

Limitations

The limitations of this version of the LDAP module are summarized in the following list.

- The module only supports authentication through a successful bind with the directory server using the FDN for this Blackboard user—the module cannot retrieve any information from the directory.
- The module only supports binding anonymously or binding with a privileged user and then performing a search for the user's FDN.

Check with Blackboard Technical Support if you have any questions regarding these limitations.

For installation problems or questions while using this document, contact Blackboard Technical Support by logging in to Behind the Blackboard at https://behind.blackboard.com. For planning, architectural analysis, best practices, or assistance with implementation, call Blackboard Technical Solutions.

LDAP Authentication

LDAP is an Internet standard that provides access to information from different computer systems and applications. LDAP uses a set of protocols to access information directories and retrieve information. A directory is like a database, but contains information that is more descriptive and attribute-based. Information in a directory is generally read more often than it is written or modified. LDAP allows an application, running on the Institution’s computer platform, to obtain information such as user names and passwords.

Centralizing this type of information is very beneficial. It simplifies the job of the administrator by providing a single point of administration. It also provides a single location for user information, reducing the storage of duplicate information. This, in turn, reduces maintenance needs. LDAP authentication also enables users to have a single login and password to access a number of different applications.

Setting Up LDAP Authentication Properties

The properties set in the `authentication.properties` file include general properties for LDAP configuration, as well as properties for individual directory servers. When adding multiple servers the variable x represents the sequence number. Parameters must be set for each directory server that Blackboard Learn will authenticate against. The LDAP module will access the servers according to the sequence number.

File Format

The `authentication.properties` and `bb-config.properties` files contain a series of properties that must be set before authentication against the Institution's directory server or servers can occur. Each property is listed with an equal sign followed by the corresponding value.

How to Edit the Authentication Properties File

Open the `authentication.properties` file in an editor and set the LDAP specific properties to match the Institution. Descriptions of the properties appear in the following section.

Properties that are suffixed with a number are properties that are associated with an individual directory server. To add information for additional directory servers, add a group of properties suffixed with the next available sequence number. The LDAP module will access the servers in the order in which they are sequenced.

LDAP Property Configuration

The table below details the LDAP properties configured through the `authentication.properties` file.

Property	Description
`auth.type.ldap.impl`	Defines the class which must conform to the `HttpAuthModule` interface. The default value, `blackboard.platform.security.authentication.LDAPAuthModule`, should not be changed unless the Institution builds and implements its own class for LDAP authorization.

Property	Description
auth.type.ldap.use_challenge	Defines the encryption setting where a value of false indicates base 64 encryption and a value of true indicates MD5 encryption. The default value is false. MD5 encryption should only be used if the LDAP servers use MD5 encryption in the same manner as Blackboard. In most cases, using base 64 encryption and securing the connection between Blackboard Learn and the LDAP servers with SSL is the best approach.
auth.type.ldap.num_servers	Defines the number of directory servers in use. For each server, there must be a corresponding set of server properties. This property must be kept current; update it each time a new server's entries are added to the authentication.properties file.
auth.type.ldap.user_not_found_fallback	Can be set to true or false. By default, this property is set to false because of security considerations. If set to true, the module will attempt to authenticate the user using the password in Blackboard Learn database if the user is not found in any of the directory servers.
auth.type.ldap.error_fallback_to_bb	Can be set to true or false. By default, this property is set to false because of security. If set to true, the module will attempt to authenticate the user using the password in Blackboard Learn database if there is an error connecting to any of the directory servers.
Server Specific Properties	
auth.type.ldap.server_url.x	The URL of the directory server including port. Example: ldap://directory.university.edu:389 If the LDAP server is setup to communicate over SSL, the URL should be: ldaps://directory.university.edu:636.
auth.type.ldap.server_ssl.x	Must be set to true or false. If set to true, the module will attempt to connect to the LDAP directory using SSL. The LDAP server must be set up to handle SSL connections.
auth.type.ldap.use_priv_user.x	Must be set to true or false. If set to true, the module will bind to the LDAP server as a privileged (specific) user when searching for the FDN of the user to authenticate.
auth.type.ldap.user_fdn.x	The user binds as this FDN. Leave as (none) if not applicable.
auth.type.ldap.user_pwd.x	The password of the privileged user. Leave as "(none)" if not applicable.
auth.type.ldap.deref_aliases.x	Set this property to configure how aliases are dereferenced during search operations. The following values are defined for this property: • always: Always dereference aliases. • never: Never dereference aliases. • finding: Dereference aliases only during name resolution (that is, while locating the target entry). • searching: Dereference aliases once name resolution has been completed (that is, after locating the target entry).

Property	Description
`auth.type.ldap.user_tag.x`	Set this property to the attribute containing Blackboard Learn User Name. This setting is domain specific.
`auth.type.ldap.server_error_fatal.x`	Must be set to `true` or `false`. If set to `true`, the module will exit with a fatal error if there is an error connecting to the server.
`auth.type.ldap.context_factory.x`	Set this property to handle password expiration warnings for LDAP accounts. The following values are defined for this property: • `blackboard.platform.security.authentication.PasswordPolicyContextFactory` for IETF-compatible LDAP servers (Novell, Active Directory). This is the default value. • `blackboard.platform.security.authentication.ResponsePolicyContextFactory` for Netscape-compatible LDAP servers supporting the Netscape response control specification.
`auth.type.ldap.referral.x`	The value of this property is a string that specifies how referrals should be handled by the module. The following values are defined for this property: • `follow`: Automatically follow any referrals. • `throw`: Throw a Java `ReferralException` for each referral. This will result in an error condition for this server. • `ignore`: Ignore referrals if they appear in results. In debug mode, a log message will be generated to indicate an incomplete result, but this will not result in an error condition for this server.
`auth.type.ldap.referral_limit.x`	The value of this property is a string of decimal digits specifying the maximum number of referrals to follow in a chain of referrals. A setting of zero indicates that there is no limit.
`base_search_fdn`	The starting point in the LDAP directory structure for searching for a Blackboard Learn user.

Example

The following is an example of the LDAP properties configured through the `authentication.properties` file.

```
auth.type.ldap.impl=blackboard.platform.security.authentication.LDAPAuthModule

auth.type.ldap.use_challenge=false

auth.type.ldap.error_fallback_to_bb=false

auth.type.ldap.user_not_found_fallback_to_bb=false

auth.type.ldap.log_level=error

# Available property values for auth.type.ldap.log_level are fatal,error,warning,information,debug

auth.type.ldap.num_servers=2
```

```
# The auth.type.ldap.num_servers property value must be increased with each
server configuration addition. If there are three server configurations, then
the value must be 3 for this parameter.
```

Server #1 Configuration

```
auth.type.ldap.server_ssl.1=false

# The auth.type.ldap.server_ssl property value sets SSL interaction between #
the Blackboard installation server and LDAP server to true or false.

auth.type.ldap.base_search_fdn.1=dc=dc,dc=blackboard,dc=com

auth.type.ldap.deref_aliases.1=never

auth.type.ldap.server_url.1=ldap://lsvr1

auth.type.ldap.use_priv_user.1=true

auth.type.ldap.user_fdn.1=CN=UserA,cn=Special
Users,dc=dc,dc=blackboard,dc=com

auth.type.ldap.user_pwd.1=test1

auth.type.ldap.user_tag.1=sAMaccountName

auth.type.ldap.referral.1=ignore

auth.type.ldap.referral_limit.1=0

auth.type.ldap.server_error_fatal.1=true

auth.type.ldap.context_factory.1=blackboard.platform.security.authentication.
PasswordPolicyContextFactory
```

Server #2 Configuration

```
auth.type.ldap.server_ssl.2=false

# The auth.type.ldap.server_ssl property value sets SSL interaction between #
the Blackboard installation server and LDAP server to true or false.

auth.type.ldap.base_search_fdn.2=dc=dc,dc=blackboard,dc=com

auth.type.ldap.deref_aliases.2=never

auth.type.ldap.server_url.2=ldap://lsvr2

auth.type.ldap.use_priv_user.2=true

auth.type.ldap.user_fdn.2=uid=UserB,ou=Special
Users,dc=dc,dc=blackboard,dc=com

auth.type.ldap.user_pwd.2=test2

auth.type.ldap.user_tag.2=uid

auth.type.ldap.referral.2=ignore

auth.type.ldap.referral_limit.2=0

auth.type.ldap.server_error_fatal.2=true

auth.type.ldap.context_factory.2=blackboard.platform.security.authentication.
PasswordPolicyContextFactory
```

Using LDAP Authentication Fail Over

Administrators must determine how Blackboard Learn should function if the directory servers are not functioning correctly at the time of an authentication request or if a user who does not exist in the LDAP database attempts to login to Blackboard Learn.

Automatic authentication fail over may be set for one or both of the following properties:

```
auth.type.ldap.error_fallback_to_bb
auth.type.ldap.user_not_found_fallback
```

Automatic fail-over functionality poses certain security risks that are discussed below in Security Considerations.

Note: The behaviors listed in the Troubleshooting LDAP Scenarios do not apply if the default configuration is changed.

Automatic Fail Over for LDAP Server Error

LDAP authentication is intended as an enterprise-level integration; therefore, the expectation is that the LDAP server will be managed administratively as a mission-critical system. The LDAP interface was developed to depend upon the constant availability of the directory servers.

Automatic fail over in the case of LDAP server error enables Institutions that are not supporting LDAP as a mission-critical system to allow users access to the system if the LDAP server fails. Automatic authentication fail over will allow Blackboard Learn to continue to run in the event that the LDAP server or servers do not function correctly. In this instance, automatic fail over is set for the `auth.type.ldap.error_fallback_to_bb` property.

Automatic Fail Over for Users who Do Not Exist in LDAP Database

This fail over option allows users who do not exist in the LDAP database to log into Blackboard Learn. Examples are administrators, users, or students who are auditing, but are not enrolled, in a class. In this instance, automatic fail over is set for the `auth.type.ldap.user_not_found_fallback` property.

Security Considerations

Automatic authentication fail over has some additional security considerations:

- ***Passwords are not synchronized***: Blackboard Learn will not know the passwords in LDAP, so Administrators have to keep track of separate passwords.
- ***Security back doors***: Automatic fail over for authentication may introduce serious security problems not related to Blackboard Learn. For example, if a user attempted a denial of service (DOS), they could shut down the directory server and attempt to log in with default passwords, which is the user name.

Synchronizing user data between the LDAP servers and Blackboard Learn (via Snapshot or the Event Driven APIs) can prevent failover from using default passwords and also enable failover to require the same password as the normal LDAP authentication.

How to Enable Authentication Fail Over for LDAP Server Error

1. Set the authentication property `auth.type.ldap.error_fallback_to_bb` to `true`.
2. Populate Blackboard Learn database with correct username and password information.
3. Restart the application server.

To learn more, see LDAP Property Configuration.

How to Enable Authentication Fail Over for Users who Do Not Exist in LDAP Database

1. Set the authentication property `auth.type.ldap.user_not_found_fallback` to `true`.
2. Populate Blackboard Learn database with correct username and password information.
3. Restart the application server.

To learn more, see LDAP Property Configuration.

Using LDAP with Active Directory

Administrators may decide to use Active Directory via LDAP. This may be done by connecting to Active Directory via an anonymous bind or by using a privileged user. The following topic explains how to set up an anonymous connection or a privileged connection, and some accompanying security risks.

Connecting via an Anonymous Bind

Active Directory does not allow anonymous access by default, but Administrators may enable anonymous searches if they choose.

Note: There are security risks with allowing anonymous LDAP binds with Active Directory; in this case, any users who have network access to the Active Directory server can search Active Directory.

How to Enable Anonymous Searches on the Active Directory Server

1. On the Windows 2000 Active Directory server, run the Active Directory Users and Groups administration tool.

2. Select the top level of the directory from the tree view in the left hand panel, and right click. Select the first item on the menu, which begins with **Delegate Control**. Click **Next**.
3. In the next window, titled **Users or Groups**, click **Add**.
4. On the next list, select **ANONYMOUS LOGON** and click **Add**. Administrators may also need to select **Everyone** and the **Guests** group, depending on how Active Directory is configured. Click **OK** when this is done. Click **Next**.
5. Select **Create a custom task to delegate** and click **Next**.
6. In the next list, select **Read**. **Read All Properties** will be selected at the same time. Click **Next**.
7. Click **Finish**.

Connecting via a Privileged Bind

By default, Active Directory can only be searched via LDAP if a privileged user is used to connect to the LDAP server. A privileged bind requires the distinguished name (DN) and password for the user. There are two options for connecting via a privileged bind:

- Create a new Windows user within Active Directory. Assign this user only the right to read access to the directory. Use this user as the privileged user.
- Use an existing Windows user as the privileged user.

Note: There are security risks with connecting via a privileged bind to Active Directory. Any user who can navigate to the file system and locate the `authentication.properties` file may find the user ID and password of the privileged user.

Troubleshooting LDAP with Active Directory

For Administrators using a Windows workstation, the LDP executable may be used to troubleshoot LDAP authentication properties. The LDP executable, found on the Windows 2003 Server CD in the `\SUPPORT\TOOLS` folder, allows LDAP operations to be performed against Active Directory and includes a graphical user interface. To learn more, see Troubleshooting LDAP.

The only change for this procedure is in Steps 2, 10 and 12. Follow the steps below when using the LDP executable against Active Directory:

1. Login as the Windows user (username, password, domain) whose username and password are being used for the privileged bind.
2. Add `sAMAccountName` to the ***Attributes*** field and click **OK**.
3. Enter `(sAMAccountName=WindowsUserName)` in the **Filter** field, where `WindowsUserName` is the Windows username that will be used as the privileged user for binding to LDAP.

For Administrators using a UNIX workstation, the LDAP Browser may be used to troubleshoot LDAP authentication properties. To learn more, see Troubleshooting LDAP.

Troubleshooting LDAP

The LDAP module should function with minimal maintenance if the `authentication.properties` file is configured properly. This topic includes information on how to troubleshoot configuring the properties files and on maintenance for LDAP authentication.

How to Debug LDAP Authentication

Administrators may debug the LDAP authentication as part of troubleshooting.

1. In the `/blackboard.config/service-config.properties` file,under `Logging Service`, make the following change:
2. `blackboard.service.log.param.logdef.default.verbosity=debug`
3. Restart the following services:
4. `/blackboard_home/tools/admin/ServiceController services.stop`
5. `/blackboard_home/tools/admin/ServiceController services.stop`
6. Login to the system again.
7. Search `/blackboard_home/logs/bb-services-log.txt` for references to `LDAPAuthModule`.
   Windows: Open the log file in a text editor and search for `LDAPAuthModule`.
   UNIX: Execute the following: `tail -f -n200 /blackboard_home/config/service-config.properties | grep "LDAPAuthModule"`

Troubleshooting LDAP Authentication Properties for Windows

For Administrators using a Windows workstation, the LDP executable may be used to troubleshoot LDAP authentication properties. The LDP executable, found on the Windows 2003 Server CD in the `\SUPPORT\TOOLS` folder, is used to search for specific data against the Active Directory and includes a graphical user interface. For users not using Active Directory, this tool may be used in the same way against other LDAP servers.

How to Use the LDP Tool

1. Go to the **Connection** menu, uncheck the **NTLM/Kerberos** check box, and select **Bind**.
2. Enter the LDAP privileged user DN in the **User** field and the LDAP password in the **Password** field.
3. Locate the **defaultNamingContext** attribute.
4. Go to the **View** menu and select **Tree**.
5. Type the **defaultNamingContext** attribute value into the **BaseDN** field and click **OK**.

6. Locate the container for user records. By default, the DN for this container starts with CN=Users. However, the user records may be located elsewhere. Try to locate the DN that contains all faculty and student user records.
7. Record the DN that contains all faculty and student user records.
8. Double-click on the tree view of this container to see all user records.
9. Go to the **Options** menu and select **Search**.
10. Customize fields in the user records returned from the search. (This step may not be necessary)
11. With the container selected, go to the **Browse** menu and select **Search**.
12. Enter the user field to search by. The user field is the user tag property, for example, (CN=jsmith).
13. Record the **distinguishedName** attribute for this user record.
14. Verify that you can find a sample user. Type the baseDN from Step 7 and (user_tag=someUserValue) where user_tag is the name of the LDAP user record field that the client expects users to enter in the Blackboard login form. (For example, if the client expects users to login by entering their email address as the 'username' in the Blackboard login form, then the user_tag should be the name of the field that stores the user's email address).
15. Update the `authentication.properties` file:
16. Set `auth.type.ldap.base_search_fdn.1` to the DN for the container for user records (See Step 7).
17. Set `auth.type.ldap.user_fdn.1` to the `distinguishedName` attribute value for the LDAP user (See Step 13).
    Windows only: If the client wants users to login to Blackboard using a Windows username, set `auth.type.ldap.user_tag.1` to `sAMAccountName`. `sAMAccountName` is the name of the Active Directory user record field that stores the Windows username.

Troubleshooting LDAP Authentication Properties for UNIX

For Administrators using a UNIX workstation, the LDAP Browser may be used to troubleshoot LDAP authentication properties. This tool may be found at http://www.iit.edu/~gawojar/ldap/.

How to Use the LDAP Browser

1. Open the LDAP browser.
2. Click **File Menu** and select **Connect**.
3. Enter the LDAP server hostname in the **Host** field.
4. Enter the port number that the LDAP server is listening on in the **Port** field.
5. Enter the base search DN in the **Base DN** field. If a privileged bind is required, uncheck **Anonymous bind**.
6. Type the privileged user DN in the **User DN** field. If **Append base DN** is checked, the Administrator only needs to add the relative DN (for example, if the base DN

is `"OU=test users,dc=blackboard,dc=com"` and the privileged user's full DN is `"CN=privldap,OU=ldap testers,OU=test users,dc=blackboard,dc=com"`, then enter only `"CN=privldap,OU=ldap testers")`.

7. Click **Connect**.
8. Click **Search** to search for a given user DN, or scroll through the list.

How to Revert to Default Authentication

To revert to the default authentication from LDAP, change `bbconfig.auth.type` to `"rdbms"`, and restart Blackboard Learn application server.

Blackboard Application Log

Blackboard Learn log records all application events handled by the Java API. Within the log the Blackboard LDAP module writes error, warning, informational, and debug messages to the `bb-services-log.txt` file.

Common Problems

The following table outlines some of the common problems that may occur when authenticating Blackboard Learn users against LDAP servers.

Problem	Action
The LDAP module loads but users cannot log in using their LDAP passwords.	Ensure that all of the users logging in have a Blackboard Learn User Name. Blackboard Learn needs a user record to associate Course and other information to the user.
An error is posted to the `bb-services-log.txt` whenever a user tries to log into the system. The module is configured to use SSL.	Ensure that the server certificate for your LDAP directory has been imported into the keystore of the JVM on Blackboard Learn application server. The JVM needs this certificate to allow SSL connections to the LDAP directory.
The LDAP module loads, but users cannot log in. Nothing is displayed in the logs, or the messages that are displayed are insufficient to diagnose the problem.	Re-run the `auth-type.properties` file and specify a `log_level` of “debug”. Log messages will generate with more detail.

LDAP Scenarios

The following table details the systems response to a number of potential LDAP situations. The default configuration of LDAP will support the set of behaviors described here.

Issue	Symptom
The LDAP server is down	Authentication should fail with an appropriate message.
The user exists in Blackboard but not in LDAP	Authentication should fail with an appropriate message.
The user exists in LDAP but not in Blackboard	Authentication should fail with an appropriate message.
The privileged user doesn't exist or has expired	Authentication should fail with an appropriate message. Blackboard Learn configuration file must be updated to proceed.
The privileged user password has changed	Authentication should fail with an appropriate message. Blackboard Learn configuration file must be updated to proceed.
There are multiple LDAP accounts for a specific user	The search domain would be restricted to a specific context within the directory tree. The first account returned will be the one used. It is the Institution's responsibility to set the `base_search_fdn` property correctly to avoid this situation.
The LDAP SSL certificate expires	Authentication should fail with an appropriate message. The LDAP SSL certificate must be updated to proceed.

How to Troubleshoot LDAP with SSL

This section explains how to troubleshoot the SSL connection between the Blackboard server and the LDAP server for clients who are using an SSL connection to secure their LDAP server.

1. Save `/blackboard_home/apps/tomcat/bin/tomcat.sh` as `tomcat.sh.prod`.
2. Enter the following command: `cp tomcat.sh.prod tomcat.sh.debug`.
3. Insert `-Djavax.net.debug=all,record,plaintext` into `tomcat.sh.debug`.
4. Go to line 207 of `tomcat.sh.debug`. Edit this line to read as follows:

```
$JAVACMD -Djavax.net.debug=all,record,plaintext $TOMCAT_OPTS $JAVA_OPTS
$MAIN start $@ \
```

5. Enter the following command: `cp tomcat.sh.debug tomcat.sh`.
6. Restart services.
7. Login with the LDAP username and password.
8. Copy the SSL-connection trace information from `/usr/local/blackboard/logs/tomcat-jvm-stdout.txt`. See the log file example below.
9. Repeat Steps 6 and 7 until debugging is complete.

10. Enter the following command: `cp tomcat.sh.prod tomcat.sh`.
11. Restart services.

Follow these instructions for debugging and clean up on Windows:

1. Save `D:\blackboard_home\apps\tomcat\conf\jk\wrapper.properties` as `wrapper.properties.prod`.
2. Copy `wrapper.properties.prod` and name the copy `wrapper.properties.debug`.
3. Insert `-Djavax.net.debug=all,record,plaintext` into `wrapper.properties.debug`.
4. Go to line 163 of `wrapper.properties.debug`.
5. Edit that line to read as follows:

```
"wrapper.cmd_line=$(wrapper.javabin) $(wrapper.java_opts) -
Djavax.net.debug=all,record,plaintext -
Djava.security.policy=="$(wrapper.tomcat_policy)" -
Djava.security.manager -Dtomcat.home="$(wrapper.tomcat_home)" -
Dblackboard.home="$(bbapp.root)" -
Dbbservices_config="$(bbapp.root)\config\service-config.properties" -
Dorg.apache.tomcat.apps.classpath="$(wrapper.class_path.apps)" -
classpath $(wrapper.class_path) $(wrapper.startup_class) -config
$(wrapper.server_xml)"
```

6. Delete `wrapper.properties`, and then copy `wrapper.properties.debug` and name the copy `wrapper.properties`.
7. Restart services.
8. Login with the LDAP username and password.
9. Copy the SSL-connection trace information from `D:\blackboard\logs\tomcat-jvm-stdout.txt`. See the log file example below.
10. Repeat until debugging is complete.
11. Delete `wrapper.properties`, and then copy `wrapper.properties.prod` and name the copy `wrapper.properties`.
12. Restart services.

Log File Example

If the SSL -connection-setup process cannot continue, the reason for the SSL connection setup failure is printed to the tomcat-jvm-stdout.txt log. After this failure appear in the log the SSL-debug output stops. There are a number of reasons why the application server may have trouble connecting to the LDAP server over SSL. The problem can be found in the SSL-debug output. Open the tomcat-jvm-stdout.txt log; go to the end of the debug output (where it gives the reason for quitting) and then scroll backwards through the output, looking for the detailed error message.

For example, in the debug output below, the end of the output shows the message "Thread-31, SEND SSL v3.0 ALERT: fatal, description = certificate_unknown". Scrolling backwards through the log, the message "out of date cert" appears before the last certificate was processed; the certificate's information shows that the certificate had expired in 2002.

The following example includes the beginning of the debug output and the last section with the error:

```
keyStore is :
keyStore type is : jks
init keystore
init keymanager of type SunX509
trustStore is: /usr/java1.3/jre/lib/security/cacerts
trustStore type is : jks
init truststore
adding as trusted cert: [

……

out of date cert: [
[
  Version: V3
  Subject: O=HC, CN=204.165.200.98
  Signature Algorithm: SHA1withRSA, OID = 1.2.840.113549.1.1.5


  Key: com.sun.net.ssl.internal.ssl.JSA_RSAPublicKey@187197
  Validity: [From: Sun Jun 18 07:16:00 EDT 2000,
              To: Tue Jun 18 07:16:00 EDT 2002]
  Issuer: O=HC, OU=Organizational CA
  SerialNumber: [    021411e9 6f9a05e1 28e9293c c80ae5b5 1166338c 1cbc0201
0c]

Certificate Extensions: 3
[1]: ObjectId: 2.16.840.1.113719.1.9.4.1 Criticality=false
Extension unknown: DER encoded OCTET string =
0000: 04 82 01 BD 30 82 01 B9   04 02 01 00 01 01 FF 13  ....0...........
0010: 1D 4E 6F 76 65 6C 6C 20   53 65 63 75 72 69 74 79  .Novell Security
0020: 20 41 74 74 72 69 62 75   74 65 28 74 6D 29 16 43   Attribute(tm).C
0030: 68 74 74 70 3A 2F 2F 64   65 76 65 6C 6F 70 65 72  http://developer
0040: 2E 6E 6F 76 65 6C 6C 2E   63 6F 6D 2F 72 65 70 6F  .novell.com/repo
0050: 73 69 74 6F 72 79 2F 61   74 74 72 69 62 75 74 65  sitory/attribute
0060: 73 2F 63 65 72 74 61 74   74 72 73 5F 76 31 30 2E  s/certattrs_v10.
0070: 68 74 6D 30 82 01 4A A0   1A 01 01 00 30 08 30 06  htm0..J.....0.0.
0080: 02 01 01 02 01 46 30 08   30 06 02 01 01 02 01 0A  .....F0.0.......
0090: 02 01 69 A1 1A 01 01 00   30 08 30 06 02 01 01 02  ..i.....0.0.....
00A0: 01 46 30 08 30 06 02 01   01 02 01 0A 02 01 69 A2  .F0.0.........i.
00B0: 06 02 01 17 01 01 FF A3   82 01 06 A0 5A 02 01 02  ............Z...
00C0: 02 02 00 FF 02 01 00 03   0D 00 80 00 00 00 00 00  ................
00D0: 00 00 00 00 00 00 03 09   00 80 00 00 00 00 00 00  ................
00E0: 00 30 18 30 10 02 01 00   02 08 7F FF FF FF FF FF  .0.0............
00F0: FF FF 01 01 00 02 04 06   F0 DF 48 30 18 30 10 02  ..........H0.0..
0100: 01 00 02 08 7F FF FF FF   FF FF FF FF 01 01 00 02  ................
0110: 04 06 F0 DF 48 30 58 A1   58 02 01 02 02 02 00 FF  ....H0X.X.......
0120: 02 01 00 03 0D 00 40 00   00 00 00 00 00 00 00 00  ......@.........
```

```
0130: 00 00 03 09 00 40 00 00   00 00 00 00 00 30 18 30  .....@.......0.0
0140: 10 02 01 00 02 08 7F FF   FF FF FF FF FF FF 01 01  ................
0150: 00 02 04 11 E9 6F 9A 30   18 30 10 02 01 00 02 08  .....o.0.0......
0160: 7F FF FF FF FF FF FF FF   01 01 00 02 04 11 E9 6F  ...............o
0170: 9A A2 4E 30 4C 02 01 02   02 01 00 02 02 00 FF 03  ..N0L...........
0180: 0D 00 80 00 00 00 00 00   00 00 00 00 00 00 03 09  ................
0190: 00 80 00 00 00 00 00 00   00 30 12 30 10 02 01 00  .........0.0....
01A0: 02 08 7F FF FF FF FF FF   FF FF 01 01 00 30 12 30  .............0.0
01B0: 10 02 01 00 02 08 7F FF   FF FF FF FF FF FF 01 01  ................
01C0: 00                                                 .


[2]: ObjectId: 2.5.29.35 Criticality=false
AuthorityKeyIdentifier [
KeyIdentifier [
0000: 01                                                 .
]

]

[3]: ObjectId: 2.5.29.15 Criticality=false
KeyUsage [
  DigitalSignature
  Key_Encipherment
]

]
  Algorithm: [SHA1withRSA]
  Signature:
0000: 50 75 22 E0 14 FE E7 50   FE 44 3B 36 D2 C8 EC 10  Pu"....P.D;6....
0010: 49 8D 48 1D 6F E6 91 1A   05 1E 8E FD 69 D3 4D 70  I.H.o.......i.Mp
0020: C3 3C FE 14 D0 D4 99 DE   CA BF 23 57 80 A0 04 F2  .<........#W....
0030: 45 33 BD B0 53 2D 72 A1   43 DD 7C 80 DD 6B 3E EC  E3..S-r.C....k>.
0040: 94 73 F9 83 21 2C 80 17   B1 CE 6E 19 FD 14 FF A8  .s..!,....n.....
0050: C0 CB 51 C7 1A C1 C0 E4   71 2F 46 9D 50 91 52 E8  ..Q.....q/F.P.R.
0060: 5B CA 24 84 FF 7F 3E 84   32 09 AA 43 66 E8 CD AB  [.$...>.2..Cf...
0070: 65 EC 5C 89 88 43 3C 15   07 3C 9D 52 AA CF 31 A1  e.\..C<..<.R..1.
0080: C9 B6 3A 7A CC 35 1B 66   CB 3C 80 00 32 15 76 2F  ..:z.5.f.<..2.v/
0090: 86 82 26 31 2F C3 EC 58   CE DD E8 E6 A4 58 6E F0  ..&1/..X.....Xn.
00A0: 70 14 36 DF CB 29 E0 E7   D4 1A 33 62 4E B7 62 3C  p.6..)....3bN.b<
00B0: 77 54 9E AA BE 57 0E 7C   F2 E1 92 D5 B0 AF E9 BB  wT...W..........
00C0: 20 CA A7 AA 4F D4 37 02   DE B2 16 9D FC 7E 90 63   ...O.7........c
00D0: 10 22 49 20 76 97 83 8A   83 0E BB A6 7B B0 E4 DE  ."I v...........
00E0: FB 62 51 FD 92 EB 9F C7   B6 91 F2 94 5C 93 29 11  .bQ.........\.).
00F0: B9 A2 AE 28 46 00 BE 14   EC 1C F8 6C 63 A3 10 BA  ...(F......lc...

]
Thread-31, SEND SSL v3.0 ALERT:  fatal, description = certificate_unknown
Thread-31, WRITE:  SSL v3.0 Alert, length = 2
```

Web Server Delegation

About Web Server Delegation Authentication

Web servers support a range of technologies for identifying and authenticating users. By default, the application server (Tomcat) handles authentication. During Web Server Delegation authentication, the application server delegates some aspects of authentication to the Web server (Apache® or IIS).

Management

When using Web server delegation much of the work related to the authentication is handled on the Web server. The process for Web server authentication is:

1. Obtain the credentials of the user.
2. Verify the user's credentials.
3. Set a header in the request that is populated with the value for the CGI variable REMOTE_USER.

If a custom module has been created, it will look for the header name that was set in Step 3. If it finds this header, the value will be used as the external user ID.

Steps 1 through Step 3 must be implemented by a Web server module/filter. For example, Blackboard provides an Active Directory filter for IIS. Clients using UNIX can set up Kerberos with Apache. The administrator must ensure that the authentication filter has been installed on the Web server. If using Windows®, see About Active Directory Authentication.

Step 4 takes place within the application server and requires that the `authtype.webserver.impl` is specified in the `authentication.properties` file.

Note: Kerberos authentication may be incompatible with direct access to WebDAV. Institutions using this type of authentication may be able to take advantage of WebDAV by first authenticating with Blackboard Learn, and then launching the Web Folder or Shared Location from within the user interface.

Implementation

The following are steps for implementing Web server delegation:

1. Install the appropriate filter (for example, Kerberos on Apache).
2. Update the `authentication.properties` file if needed.
3. Restart the application server and the Web server.

Web Server Delegation with Windows 2003

It is necessary to complete some additional steps to use Web Server Delegation with Windows 2003. Note that Active Directory authentication is a form of Web Server Delegation.

How to Configure Web Server Delegation with Windows 2003

1. Edit the `authentication.properties` file. To learn more, see Active Directory Property Configuration.
2. Edit the `bb-config.properties` file to change `bbconfig.auth.type` to `bbconfig.auth.type=webserver`.
3. Deploy the configuration updates by running `cd BB_DEPLOY_DIR\tools\admin PushConfigUpdates.bat`. Running `PushConfigUpdates` will overwrite any customizations to the configuration. To learn more, see Appendix A: PushConfigUpdates.
4. Run `websitereinstall.bat`, which will delete and recreate the Blackboard website. All custom changes to IIS will be removed, including configurations to support SSL Choice.
5. Open IIS 6.0.
6. Right click on the Blackboard website and click **Properties**.
7. In the IIS Management Console for Blackboard Learn, select the authentication method or methods. The options are **Basic**, **Digest**, and **Integrated Windows** authentication. One or more options may be selected.
8. Click the **ISAPI Filters** tab. Verify that the Windows `.dll` has been added. Also, verify the order of the filters. They should be in the following order: Sessiontracker, Windows, Jakarta. If they are not in order, move them into the correct order.
9. Click **Apply**.
10. Click **OK**.

Active Directory

About Active Directory Authentication

Blackboard Learn supports Web server delegated authentication. It includes a Web server filter for Active Directory that will authenticate users against an Institution's Active Directory server. All the files necessary to support Active Directory authentication are included with Blackboard Learn.

Active Directory Authentication

Microsoft® Active Directory™ is a part of the Microsoft® Windows® network architecture and is intended as an enterprise-level integration. It allows Organizations to share and manage information about network resources and users. In addition, Active

Directory acts as the central authority for network security, letting the operating system verify a user's identity and control his or her access to network resources, such as data, applications, or printers. There are a number of benefits to using Active Directory. Administrators have a single point of management for Windows-based user accounts, clients, servers, and applications. Active Directory authentication also allows for standardized business rules for applications and network resources.

Security Considerations

When the Active Directory authentication is implemented a user has a single Microsoft Windows User ID and a single Blackboard Learn User Name. The database has a single entry for this user. The level of security for the Windows User ID and password is the same as that for Blackboard Learn User Name and password.

When users are working on a shared computer they must close the browser and log out of Windows at the end of the session to ensure their security in maintained.

Limitations

The user's Blackboard Learn User Name is associated with their Microsoft Windows' login. This means that the user's database record is linked to their Windows User Name and password allowing them to only have one Blackboard Learn User Name. For example, Windows user "jdoe" can only have one Blackboard Learn login; he or she cannot be "Instructor1" and "Student2" in Blackboard Learn.

To ensure the safety of user accounts, the browser must be closed and the user must log off of Windows when a session on a shared computer is ended.

For questions about these limitations, contact Blackboard Technical Support by logging in to Behind the Blackboard at https://behind.blackboard.com.

Active Directory Authentication and Portal Direct Entry

Blackboard does not currently support using Web Server Delegation with Portal Direct Entry. Clients who would like to set up a customized authentication with Web Server Delegation and Portal Direct Entry should contact Blackboard Global Services.

Setting Up Active Directory

The properties set for Active Directory authentication are found in the `auth.type.webserver` section of the `authentication.properties` file.

Note: There is no special configuration needed for IIS.

File Format

The `authentication.properties` file contains a series of properties that must be set before authentication against the Institution's Active Directory server or servers can occur. Each property is listed with an equal sign followed by the corresponding value.

How to Set the Authentication Type

Prior to editing the `authentication.properties` file, the authentication type `(bbconfig.auth.type)` must be edited in the `bb-config.properties` file. This allows Blackboard Learn to select the appropriate set of `auth.type*.*` entries at start up. The following steps are instructions for setting the authentication to Active Directory.

1. Edit the `authentication.properties` file.
2. Edit the `bb-config.properties` file so that `bbconfig.auth.type=webserver`.
3. Run `PushConfigUpdates` to activate the changes. To learn more, see Appendix A: PushConfigUpdates.
4. IIS only: Run the Website Reinstaller tool to remove and reinstall IIS: `blackboard_home\tools\admin\WebsiteReinstall.bat`
   In the IIS Management Console for Blackboard Learn, select the authentication method or methods. The options are Basic, Digest, and Integrated Windows authentication. One or more options may be selected.

Note: All custom changes to IIS will be removed after running this tool. For example, configurations to IIS to support SSL Choice will be erased.

Active Directory Property Configuration

The table below details the properties configured through the `authentication.properties` file.

Property	Description
`auth.type.webserver.impl=blackboard.platform.security.authentication.WindowsAuthModule`	Defines the class which must conform to the `HttpAuthModule` interface. The default value, `blackboard.platform.security.authentication.WindowsAuthModule`, should not be changed unless the Institution builds and implements its own class for Web Server Delegation authorization. If using Kerberos or another type of Web Server Delegation and it is configured to set the standard REMOTE_USER header then use `blackboard.platform.security.authentication.ExternalAuthModule`.
`auth.type.webserver.user_account`	Describes how external users are handled by Blackboard Learn. It is set to reconcile, create or deny. The first thing the system does for any of these settings is check for an existing user account that is associated with the external User ID. If one is not found, the following will occur: If set to `reconcile` the system will display a page that allows the user to login as an existing Blackboard user once, associate that Blackboard user account with the external user ID, and then will log them in via Web server delegated authentication in the future. If set to `create`, the system will try and create a new user account with minimal information that has to be updated by the user or Administrator. The user name is automatically webserver-user-user number (for example, webserver-user-100). If set to `deny`, a message will be displayed for the user to contact the

Property	Description
	Administrator.
`auth.type.webserver.suppress`	This property, if set to `true` hides the login form fields. This should be used when an external authority is the only place where passwords should be entered, as is standard with web-based authentication.

Example

The example below details the properties configured for Web server delegation through the `authentication.properties` file. The example uses reconcile to handle external users. Create and Deny are also valid options for this property.

```
auth.type.webserver.impl=blackboard.platform.security.authentication.WindowsAuthModule

auth.type.webserver.user_account=reconcile

auth.type.webserver.suppress=true
```

Customized Authentication

About Custom Authentication

Custom authentication enables developers to create a customized authentication module that may be plugged into the authentication framework of Blackboard Learn. Blackboard Learn ships to clients with several authentication techniques, each with its own associated module.

Developing customized authentication allows clients to do either of the following:

- Replace the pre-built modules with a module provided by the client or by a third party.
- Specify a different authentication technique with its own properties. Any module that conforms to the Blackboard End-User Authentication API may be substituted.

The Blackboard Challenge-Response authentication type is installed by default, and will be referred to as "the default authentication type" throughout this section. Module refers to a Java class that implements the interface `blackboard.platform.security.authentication.HttpAuthModule`. To learn more about this interface, see About Authentication through the API.

Note: For questions about creating highly customized authentication implementations, please contact Blackboard Technical Solutions.

Audience

Developers who wish to create a customized authentication module should have the following:

- A good understanding of general security principles, especially regarding authentication and Web based security.
- Experience with Java servlet programming.

Data Model

There is no dedicated data model for authentication. However, the default authentication process relies on the password and user name attributes of the user entity. For details about the data model, see the Blackboard Building Blocks Extension Developer's Guide.

Setting Up and Deploying Custom Implementations

This section explains how developers may use the implementations provided by Blackboard Learn to create custom authentication modules for their Institution. It also reviews how to extend Blackboard Learn default implementation for a custom

authentication module and how to create custom authentication modules for LDAP and Web server delegation.

Extending Blackboard-provided Implementations

Rather than creating a custom implementation of the `HttpAuthModule` interface from scratch, clients may create a Java class that inherits from one of the `HttpAuthModule` implementations included with Blackboard Learn. Re-using existing application components allows for custom authentication functionality with a minimum of development effort.

Extending the Blackboard Default Implementation

The default implementation may be re-used to simplify implementations that only require custom processing of the user credentials. This has the added benefit of being able to re-use the challenge/response features of the default implementation without duplicating code.

The simplest way to extend the default implementation is to extend the class `BaseAuthenticationModule`, over-riding only the `doAuthenticate()` method.

The `doAuthenticate()` method returns a string representing the username for the user who has been authenticated. Subclasses of `BaseAuthenticationModule` may either completely over-ride the `doAuthenticate()` method, or re-use the result. The following is an outline of an implementation that re-uses the result:

```
import blackboard.platform.security.authentication.*;
public class CustomAuthModule extends BaseAuthenticationModule
  throws BbSecurityException, BbAuthenticationFailedException,
BbCredentialsNotFoundException
{
  public String doAuthenticate(HttpServletRequest request,
                HttpServletResponse response)
 {
 // Get the default authentication technique's result
 String username = super.doAuthenticate( request, response );
 /* OUTLINE OF POSSIBLE CUSTOM AUTH CODE */
 // Check some data source external to Blackboard Learn
 // Do some extra processing related to custom authentication check

 // Return the username if no exceptions have been thrown
 return username;
 }
```

```
}
```

Alternatively, any or all of the methods in the `HttpAuthModule` interface may be overridden in custom authentication modules.

Creating a Custom LDAP Implementation

The Blackboard `LDAPAuthModule` is a very straightforward implementation of LDAP authentication. Custom LDAP implementations that wish to extend the functionality of the Blackboard `LDAPAuthModule` by binding to a given LDAP schema and performing queries specific to that schema can do so by subclassing from the Blackboard class and overriding its `authenticate()` method. The LDAP module may be extended to add additional behavior (such as additional processing after calling `super.authenticate()`). However, changing the behavior of the LDAP module cannot be overridden. If the Blackboard-supplied LDAP module is not sufficient, a completely new module (extending `BaseAuthenticationModule`) could be developed to access the LDAP directory in a specialized way

For a detailed discussion of the property settings for LDAP authentication, see Setting Up LDAP Authentication Properties.

Creating a Custom Web Server Delegation Implementation

For custom authentication implementations that rely on Web server modules or filters, `ExternalAuthModule` has been provided as the simplest possible implementation. The `ExernalAuthModule` method parses the request and extracts the CGI variable `REMOTE_USER` as the user name to authenticate against. The custom Web server delegation implementation would only need to install a module or filter (for example, Kerberos on Apache) on the Web server and configure it to populate the request with the CGI variable `REMOTE_USER`.

Deploying Custom Implementations

Once the .JAR file containing the .class file for the custom authentication class is built, place the .JAR file in the tomcat auto load directory.

The custom .JAR files are stored in the common classloader of Tomcat. Follow the steps below:

1. Open the directory `config/tomcat/classpath` and create a new text file named `yourinstitution-common.classpath.bb`.
2. Open this text file and enter the full name of the .JAR files that should be included in the classpath.
3. Run `PushConfigUpdates` to copy the file to the correct location. To learn more, see Appendix A: PushConfigUpdates.

Note: A copy of the jar file should be in `/systemlib`. Additionally, edit `/system/build/bin/launch-tool.bat` (or .sh on Unix) and append the .jar files to the `BB_CP` variable. Otherwise, command line tools that bootstrap the core services (for example, `LogRotation` or `PurgeAccumulator`) will not work.

Updating the Collaboration Server

Administrators who use custom authentication may experience issues with the Collaboration server. The collaboration server breaks because of the AccessManager in `service-config-collab-server.properties`. This issue may be resolved by adding the custom authentication module class to the collaboration server's classpath. Follow the steps below:

How to Update the Collaboration Server in Windows

1. Add the classpath to the location of the authentication module classes (either a path to a directory, or a path to a jar file) to the `install-nt-services.bat.bb`. Administrators may use the wildcard syntax for the Blackboard install directory.
2. Run `PushConfigUpdates` to activate the changes. To learn more, see Appendix A: PushConfigUpdates.

Note: When updates to the system are installed, the .bb file may be overwritten. Save a copy of the .bb file before an update and use this copy to replace the .bb template.

How to Update the Collaboration Server in UNIX

1. Add the classpath to the location of the authentication module classes (either a path to a directory, or a path to a jar file) to the `collabserverctl.sh.bb`. Administrators may use the wildcard syntax for the Blackboard install directory.
2. Run `PushConfigUpdates` to activate the changes. To learn more, see Appendix A: PushConfigUpdates.

Note: When updates to the system are installed, the .bb file may be overwritten. Save a copy of the .bb file before an update and use this copy to replace the .bb template.

Updating the launch-tool Script

Administrators should add the custom authentication jar file to the classpath of the `launch-tool` and `launch-app` scripts to prevent issues with the `PurgeAccumulator` tool and other administrative tools.

How to Update the launch-tool Script in Windows

Add the classpath to the location of the authentication module classes (either a path to a directory, or a path to a jar file) to the `launch-tool.bat`. This file is located in `blackboard\system\build\bin`.

Note: Save a copy of the files before attempting to add the classpath.

How to Update the launch-tool Script in UNIX

Add the classpath to the location of the authentication module classes (either a path to a directory, or a path to a jar file) to the `launch-tool.sh`. This file is located in `blackboard/system/build/bin`.

Note: Save a copy of the files before attempting to add the classpath.

Using WebDAV with a Custom Implementation

For WebDAV access to the Content Collection to function correctly when a Custom Authentication Module is deployed, the module must implement `IUserPathAuthModule` for LDAP implementations or `ExternalAuthModule` for all other custom implementations. For information about implementing `IUserPathAuthModule`, see Planning Authentication Implementation.

Troubleshooting Custom Implementations

This section explains how to troubleshoot problems that may occur when installing a custom authentication module.

- Examine the tomcat logs in `\blackboard\logs\tomcat` and `\blackboard\logs\bb-services-log.txt` files for Java errors that may occur.
- Ensure that the class name is correct.
- Ensure that all of the properties are correctly defined in the `authentication.properties` file.
- Ensure that the authentication module (and all of its dependencies) are in the .JAR file and are in the proper location.
- If subclassing `BaseAuthenticationModule` or other implementations, make sure that there are no namespace conflicts that may affect processing.

Note: Session expiration logging has the potential to cause performance issues in log searched for large installations.

Setting Up Customized Authentication Page Flow

This section discusses how to customize the routing between pages in Blackboard Learn. An Institution may customize routing by uploading a custom login page to Blackboard Learn server (via the **Customize Login Page** link on the System Control Panel), or by implementing the `requestAuthenticate()` method on the `HttpAuthModule` interface.

The **Customize Login Page** function on the System Control Panel allows the Administrator to download a template for the login page and then upload a modified

template to the server. This allows the Administrator to add extra script functionality to the login page hosted by Blackboard Learn.

Note: Institutions interested in customizing their Blackboard Learn login page must not remove the JSP tags on the page.

Implementing requestAuthenticate()

If the user wishes to redirect to a login form on a page hosted by another application, instead of using the **Customize Login Page** function, the user should implement the `HttpAuthModule` interface method `requestAuthenticate()` to do a redirect.

For authentication to function properly, any login form on a page hosted by another application must submit the form to the login broker at the Institution's URL (for example, a login page hosted at "http://another.institution.com" must submit its login form to the URL "http://your.institution.edu/webapps/login").

Redirecting to the Original Target URL

If a user has clicked a bookmarked URL that leads into the Blackboard Learn, but they are not currently authenticated, the application will route the user to the login broker URI with the originally requested URL preserved. The login broker expects that the rest of the application will preserve the originally-requested URL, in URL-encoded form, as either a hidden form variable or a query string parameter named `new_loc`. Any custom Login page uploaded to the Blackboard Learn server, or any third-party script page that `requestAuthenticate()` redirects to, must keep this contract as well. If not, the Blackboard Learn will route to its default entry page.

About Authentication through the API

This section explains the different methods that require implementation for the authentication process. The Authentication API is defined by the Java interface `blackboard.platform.security.authentication.HttpAuthModule`. All custom authentication modules must implement this interface.

To learn more about the arguments and returns for each method, see the API Specifications.

Authentication Processing Methods

The following authentication processing methods are implemented in the authentication module. See the sample authentication module below for an example of the usage of these methods.

init()

`init()` is called by the authentication framework when it creates an instance of `HttpAuthManager`, which in turn creates an instance of the appropriate authentication module and allows the implementation class to perform any required initialization. The sole argument passed in is the `ConfigurationService` object created during system startup. This method is intended to cache properties relevant to custom authentication that are defined at system startup. Installation specific properties can be obtained from the `ConfigurationService` and is different from the properties passed in `setConfig()`. See the following method example.

An authentication module that needs to insert a completely new set of entries in the `authentication.properties` must implement an `init()` method that calls the `configure()` method on the class `HttpAuthConfig`. To learn more, see Configuration File Processing Methods.

```
    /*
     * Module initialization. This method gets called when Tomcat starts up.
     */
       public void init(ConfigurationService arg0) throws
        IllegalStateException {
        //Although this is not necessary for subclasses of
        //BaseAuthenticationModule, it is a good practice to do this
        //unless there is a reason not to.
        super.init(arg0);
        try {
        // Set up logging
```

```
        _logger = BbServiceManager.getLogService();
      } catch (RuntimeBbServiceException e) {
        e.printStackTrace();
       }
      if (_logger == null) {
         // This println statement will output to the
        //stdout-stderr.log file.
        System.out.println("logger is null");
      } else {
        _logger.logWarning("Custom Auth Module: init()");
      }
  }
```

requestAuthenticate()

`requestAuthenticate()` is called if the user is not logged into the Blackboard Learn. The implementation may prompt the user for their credentials. This method may be used for the following:

- to generate a HTTP-302 status in the response, or Javascript, to redirect to a login URL
- to generate a login form programmatically via the servlet API
- to generate an HTTP-401 response
- to forward the user to the appropriate point in the customized Blackboard Learn (for example, by using a `javax.servlet.RequestDispatcher.forward()`)

```
      /*
       * Gets called when the user needs authenticating.
       */
     public void requestAuthenticate(
      HttpServletRequest request,
      HttpServletResponse response) throws BbSecurityException {
      //Do the superclass implementation, redirect to the Default Login
      //Page
      _logger.logWarning("Custom Auth Module: requestAuthenticate()");
      super.requestAuthenticate(request, response);
     }
```

doAuthenticate()

`doAuthenticate()` is called for the implementation to:

- parse the request
- extract any credentials
- perform the authentication work

```
/*
 * Does the work of authenticating the user.
 */
  public String doAuthenticate(
   HttpServletRequest request,
   HttpServletResponse response)
   throws BbSecurityException,
     BbAuthenticationFailedException,
     BbCredentialsNotFoundException {
   //Do custom processing here, just make sure
   //the user name is returned or the appropriate exception
   //is thrown.
   _logger.logWarning("Custom Auth Module: doAuthenticate()");
   return super.doAuthenticate(request, response);
  }
```

doLogout()

`doLogout()` is called when the user explicitly logs out. The implementation may perform any tasks required of its authentication authority and remove any session variables. Custom authentication modules that subclass `BaseAuthenticationModule` may use its implementation to clean up Blackboard Learn specific session information.

```
   /*
    * Gets called when the user explicitly logs out.
    */
    public void doLogout(
   HttpServletRequest request,
   HttpServletResponse response)
   throws BbSecurityException {
     _logger.logWarning("Custom Auth Module logging out");
```

```
        //If subclassing BaseAuthenticationModule, the superclass
        //implementation can be called. It basically performs the
        //work of invalidating a user's session.
        SessionStub sessionStub = new SessionStub( request );
        sessionStub.disassociateCurrentSessionAndUser();
      }
```

Note: This method may not be called since users may close their browser and not explicitly log out.

isAuthenticated()

`isAuthenticated()` has been deprecated and is only included to maintain backward compatibility. If `BaseAuthenticationModule` is being extended, no implementation is necessary.

Note: Implementing this method will not over-ride the login broker's internal checks that determine whether the current Blackboard Learn session is authenticated.

Of the above interface methods, the key methods for cooperating with the login broker are `doAuthenticate()`, `requestAuthenticate()` and `doLogout()`.

Configuration File Processing Methods

The authentication framework expects to find an `authentication.properties` file in the `/blackboard/config` directory of Blackboard Learn. The class `blackboard.platform.security.authentication.HttpAuthConfig` loads the `authentication.properties` settings into memory and manages the process of configuring an `HttpAuthModule` object with the appropriate settings.

If an Institution wishes to create a custom authentication module that requires a completely new set of entries in the authentication properties file, then all three `HttpAuthModule` methods described below must be implemented by the custom authentication module.

getAuthType()

`getAuthType()` returns a String identifier for the authentication technique. The four authentication techniques supported by the Blackboard Learn (Blackboard default authentication, LDAP, and Web server delegation) are represented as "rdbms", "ldap", and "webserver". The authentication framework loads this String identifier from the `bb-config.properties` setting for `bbconfig.auth.type`. The authentication framework then requests that the `HttpAuthConfig` class, which has loaded all settings found in the `authentication.properties` setting, creates an `HttpAuthConfig` instance which stores all `authentication.properties` settings

with the given `auth.type` identifier. For example, if `bbconfig.auth.type` is set to ldap, then all property settings that match `auth.type.ldap` are loaded into a new `HttpAuthConfig` instance. This method should be implemented for every authentication module class. For example, with the Sample Custom Authentication Module, the following entry needs to be changed in the `bb-config.properties` file:

```
auth.type.custom.impl=blackboard.authentication.test.CustomAuthModule
```

getPropKeys()

`getPropKeys()` returns a String array of the keys to an authentication module's configuration properties. For example, the `BaseAuthenticationModule` has the keys “impl” and “use_challenge”. This method should be implemented for every authentication module class. The array must contain all of the property keys for the custom implementation. At a minimum, the method should return the “impl” property which specifies the fully qualified class name of the `HttpAuthModule` implementation. In order to obtain the value of the properties, the authentication module can use the `HttpAuthConfig.getProperty()` method with the property name. Only the property name need be specified, not the entire entry in the `authentication.properties` file. For example, with the Sample Custom Auth Module, the following entries need to be added to the `authentication.properties` file:

```
auth.type.custom.impl=blackboard.authentication.test.CustomAuthModule

auth.type.custom.prop1=test
```

To get the property of prop1, the following method can be used:

```
// _config has already been defined as a member variable of type
HttpAuthConfig

String prop1 = _config.getProperty(“prop1”);
```

setConfig(HttpAuthConfig config)

`setConfig(HttpAuthConfig config)` implies a contract between `HttpAuthModule` and `HttpAuthConfig`, such that `HttpAuthConfig` is expected to supply the correct object for a given property key. Subclasses of `BaseAuthenticationModule` do not need to implement this method, and can use the `_config` member variable to obtain configuration properties.

Using Servlet Authentication

The servlet that initializes Blackboard Learn calls `HttpAuthManager.init()`, which does the following:

- Loads the authentication configuration settings
- Creates and initializes an instance of an `HttpAuthModule` implementation class
- Installs the instance into the `HttpAuthManager` class

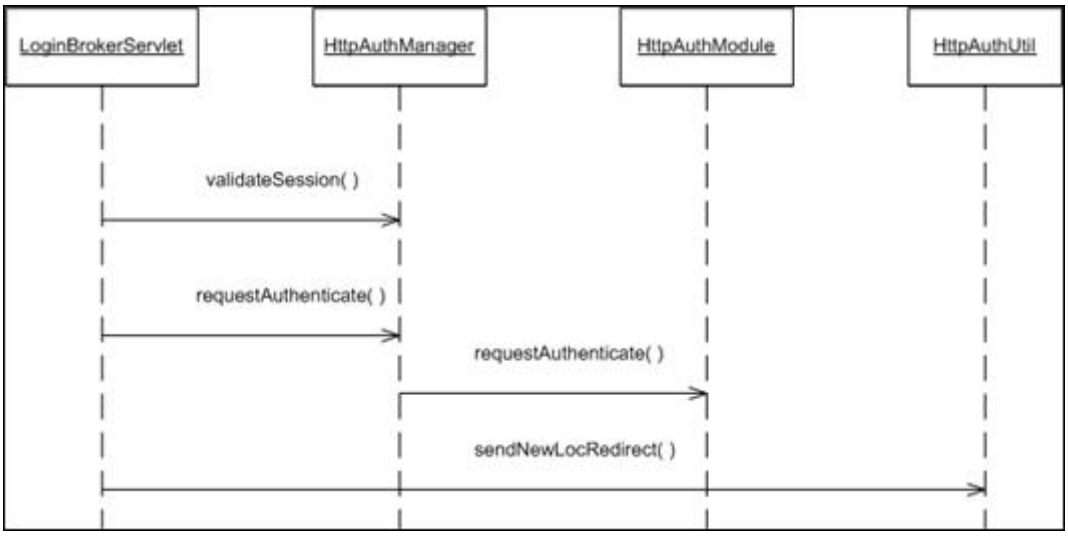
Login

Authentication processing in the Blackboard Learn is centralized in a login-broker servlet, installed at `/webapps/login`. The login-broker servlet processes all login requests for the Blackboard Learn. Login request universal resource identifiers (URIs) take the form of `/webapps/login`.

Every JSP page in Blackboard Learn that requires an authenticated user session does one of the following:

- Redirects to the login broker
- Displays a link to the login broker if the current user is not logged in or is not logged in as a user with sufficient authorization for the page

The following image demonstrates how the login-broker servlet processes login requests.



The login-broker servlet invokes the method `validateSession()` on a `HttpAuthManager` instance. The `HttpAuthManager` method `validateSession()` calls the `doAuthenticate` method on the module installed on the `HttpAuthManager` class. The servlet uses the boolean result of this method to determine whether to invoke the method `sendNewLocRedirect()` on the `HttpAuthUtil` class, or to trigger the `requestAuthenticate()` method on `HttpAuthManager` object.

The `sendNewLocRedirect()` method on the `HttpAuthUtil` class retrieves the `new_loc` parameter in the request, translates it to a URL and forwards the request to that URL.

The `requestAuthenticate()` method on the `HttpAuthManager` calls the `requestAuthenticate()` method on the module that was installed during system startup. The module's `requestAuthenticate()` method is called so the implementation may prompt the user for credentials. These credentials must then be submitted to the login broker via HTTP.

Logout

The login-broker servlet also processes all logout requests for the Blackboard Learn. Logout request URIs take the form of `/webapps/login?action=logout`.

The servlet invokes the method `invalidateSession()` on a `HttpAuthManager` instance. After the session is invalidated, the servlet forwards the request to the index page Blackboard Learn.

When a user's session has timed out, and the user tries to access a page in the Blackboard Learn that requires an authenticated user session, the system redirects the user to the login broker.

About SSL and SSL Choice

Secure Sockets Layer (SSL) is a protocol for protecting Internet communications. SSL ensures that a communication is not read or changed by another entity. Blackboard Learn uses SSL to secure all or some communications between the Web server and the client machine. This feature that allows Administrators to select which areas of Blackboard Learn are secured using SSL is called SSL Choice.

Note: SSL may also be used to secure the connection between Blackboard Learn and a separate server for authentication (such as an Active Directory server). If SSL will be used both for connecting to an authentication server and for client sessions, SSL for the authentication server must be configured first. To learn more about configuring SSL for securing with an integrated authentication server, see Using Blackboard Learn Authentication.

Note: SSL Offloading is supported as of Blackboard Learn 9.1 Service Pack 8. See section below for more information.

How Does SSL Work?

SSL works through public key encryption. Transmissions are decrypted and encrypted using certificates. The steps below outline the process for establishing a connection over SSL:

1. Client contacts the server with a list of encryption methods.
2. The Server returns its certificate and a public key. These initial communications are scrambled with random data.
3. Client validates the certificate.
4. Client creates a secret string using an encryption method recognized by both the client and the server. The string is combined with the server's public key and sent back to the server.
5. Both the client and server create session keys based on the secret string.
6. The client sends a message to the server that it will now use the session key to encrypt and decrypt communications.
7. The server responds that it will also use the session key.
8. After each side confirms, the session keys are used to encrypt and decrypt communications during the session.

How to Obtain a Certificate

The simplest way to obtain a certificate for use with a Web site is through a vendor known as a Certifying Authority (CA). The process, shown in the steps below, is relatively simple.

1. Generate a certificate request.
2. Send the request to a CA.
3. The CA creates and registers a certificate.

4. Make this certificate available to the Web Server (IIS or Apache).

Certificates created in this way are usually registered and good for one year. After one year the certificate will no longer work and a new certificate must be obtained.

To remain secure, Blackboard recommends certificates with RSA key sizes at least 2048 bits in length. As per the National Institute of Standards and Technology (NIST) guidelines for Key Management (SP 800-57), Table 4 for recommended algorithms and minimum key sizes, certificates with RSA key sizes at or under 1024 bits are no longer considered secured and a minimum 2048 bits is considered secure through 2030.

Note: If using a self-signed certificate, the certificate must be added to the list of allowed certificates on the client machine. If this is not done, the multi-upload feature will fail, as will a few other features that use SSL.

How Does SSL Appear to Users?

SSL works with the Hypertext Transfer Protocol (HTTP) to secure connections between Blackboard Learn Web server and the client machines. It is fairly easy to see when a Web page is using SSL to secure transmissions because an “s” is appended to the http at the beginning of the address.

Without SSL: http://blackboard.yourinstitution.com

With SSL: https://blackboard.yourinstitution.com

It is important to understand that if SSL is used to secure the Web page in this example then the first URL (without SSL) is invalid and will return a 404 error.

SSL Choice

The SSL Choice feature is available in the user interface from the System Control Panel. It allows an institution to decide if all, none, or some of Blackboard Learn is secured with SSL. If SSL is to be used, it is recommended and most effective when applied to the entire Web site and not just selected areas.

Note: SSL must be configured on the Web Server before using the SSL Choice feature. If SSL Choice is turned on before the Web server is configured then any areas set to use SSL will be unavailable to users!

SSL Offloading

SSL offloading relieves a web server of the processing burden of encrypting and decrypting traffic sent via SSL. If you have a system on your network that handles SSL Offloading, follow these steps to configure Blackboard Learn to make use of SSL offloading.

Note: Ensure that Learn's HTTP port cannot be accessed directly from outside the firewall.

From the Administrator Panel

Use the following instructions if you are logging in from the Administrator Panel:

1. Log into Learn as an administrator.
2. Go to Admin > SSL Choice.
3. Enable SSL system-wide.

From the Command Line

Use the following instructions if you are logging in using the command line:

```
# Enable offloading
bbconfig.appserver.ssl.offloaded=true


# This should only be set if your load balancer cannot pass the client's Host header through.
# See discussion below for more information
bbconfig.appserver.ssl.offloaded.hostname=www.example.edu
# This is the port number that Learn uses when building URLs to send to users.
# It should match the port number that your load balancer is listening on.
bbconfig.webserver.ssl.portnumber=443
```

Run `PushConfigUpdates` to activate the changes. To learn more, see Appendix A: PushConfigUpdates.

```
$ /usr/local/blackboard/tools/admin/PushConfigUpdates.sh
```

Configuring the Load Balancer

Configure your load balancer or SSL accelerator to decrypt the SSL requests and send them through to the Learn HTTP port.

Hostname Header

Some features within Learn require that you know the hostname specified by the user. The best way to accomplish this is to configure your load balancer to pass the user's requested Host header through to Learn.

If your load balancer does not support forwarding the Host header, you will need to hard-code the hostname within the Learn `bb-config.properties`. Hard-coding means that Learn features, such as hostname-based branding, will not work. To hard-code the hostname, set the `bbconfig.appserver.ssl.offloaded.hostname` property.

Client IP Address

You should configure your load balancer to inject an X-Forwarded-For header containing the client's IP address. Any tool within Learn that makes use of client IP addresses (Assessment IP restrictions, Session Fingerprinting, and so on) will then use the value loaded from this header.

If your load balancer does not support injecting this header, it is likely that the load balancer's IP address will be used for client operations within Learn. Some Learn features will not work properly.

Setting Up SSL for IIS

To use SSL to secure Blackboard Learn the IIS Web server must first be set to use SSL. Configuring SSL should only be done by an experienced Microsoft administrator.

Once SSL is configured, the SSL Choice feature (accessible from the Administrator Control Panel) will function correctly. Trying to use the SSL Choice feature before configuring SSL for Apache can result in serious system errors.

How to Configure SSL for IIS

1. Open the **Internet Services Manager**.
2. Right-click on the `blackboard_bblearn` Web site and select **Properties** from the menu.
3. Click the **Directory Security** tab.
4. Click **Server Certificate** in the Secure communications frame at the bottom of the tab.
5. The Web Server Certificate Wizard will appear. **The Status of your Web server** should report that there is not a certificate installed and there are no pending requests. If anything else appears, there may be a certificate installed or a pending request already. Click **Next** to advance.
6. Select **Create a new certificate** and click **Next** to advance.
7. Select **Prepare the request now**, but send it later and click **Next** to advance.
8. Enter a name for the certificate (the name of the Web site in IIS is the default) and select a bit length from the drop-down list. Blackboard recommends a bit length of 2048. Visit How to Obtain a Certificate for more information on RSA key size recommendations. Click **Next** to advance.
9. Enter the name of your **Organization** and your **Organizational unit** in the fields. This information is important to ensure that your certificate is unique and easily identified. Click **Next** to advance.
10. Enter the **Common name** of the Web site. The host plus the domain name works best (example: `blackboard_server.yourinstitution.edu`). Click **Next** to advance.
11. Enter the appropriate geographical information for your institution. Click **Next** to advance.
12. Enter a file name for the certificate request or click **Next** to select the default and advance.
13. Click **Finish** to create the certificate request.
14. Send the certificate request to a Certifying Authority. There are several commercial vendors or you can sign your own if you have the capability. The output from the Certifying Authority will be a file with the extension `.cer`.
15. Once you have obtained a `.cer` file, return to the Web Server Certificate Server as described in Steps 1-4.
16. Select **Process the Pending Request** and click **Next** to advance.
17. Enter the location of the `.cer` file and click **Next** to advance.

18. Click **Next** to advance through the summary steps (be sure to review the summaries to make sure you are installing the correct certificate!).
19. Return to the **Properties** box for the `blackboard_bblearn` Web site as described in Steps 1 and 2.
20. If the Web Site tab is not active, select it.
21. Enter 443 for the **SSL Port** in the **Web Site Identification** frame at the top of the tab.
22. Blackboard recommends the use of strong encryption. Typically, strong encryption means only using SSLv3 and TLSv1 algorithms. Visit the Microsoft Support article How to disable PCT 1.0, SSL 2.0, SSL 3.0, or TLS 1.0 in Internet Information Services.
23. Restart the server to complete the process.

Setting Up SSL for Apache

To use SSL to secure Blackboard Learn the Apache Web server must first be set to use SSL.

Note: Successful completion of this process requires that Solaris users are running Solaris 10, Solaris 9, or Solaris 8 with patch 112438-02.

Configuring SSL should only be done by an experienced administrator.

Once SSL is configured, the SSL Choice feature (accessible from the Administrator Control Panel) will function correctly. Trying to use the SSL Choice feature before configuring SSL for Apache can result in serious system errors.

How to Configure SSL for Apache

1. Login to the Web/application server as root.
2. Set the `PATH` to include the `OpenSSL` provided by Blackboard with the following commands:

```
PATH=/blackboard_home/apps/openssl/bin:$PATH
export PATH
```

3. Test that `OpenSSL` is in the `PATH` by executing `openssl`. If `OpenSSL` is set in the `PATH` correctly, an `OpenSSL>` prompt will appear. Enter 'q' to exit the prompt. If another instance of `openssl` is installed on the operating system make sure that the version supplied by Blackboard is the version that appears in the `PATH`.
4. Create a directory to store certificates. Then change directories. For example:

```
mkdir /blackboard_home/apps/httpd/conf/certs/
cd /blackboard_home/apps/httpd/conf/certs/
```

5. Create a RSA private key:

```
openssl genrsa -out server.key 2048
```

   where server is a variable for the file name. Typically the server name is used. Visit How to Obtain a Certificate for more information on RSA key size recommendations.

6. Backup this file and make sure that only root has read permissions on it. Make sure that the password is secure and can be recalled when necessary. (need to recall to start the server).
7. Create a Certificate Signing Request (CSR) for the server RSA private key with the following command:

```
openssl req -new -days 365 -key server.key -out server.csr
```

The –days option sets the expiration of the certification. Most Certifying Authorities will only sign a certificate for 1 year. At that time the certificate must be resigned.

8. View the details of the CSR with the following command:

```
openssl req -noout -text -in server.csr
```

When submitting the request, it may be necessary to view the file and copy text from it for submission to the Certifying Authority (CA).

9. Send the CSR to a Certifying Authority for signing. There are several commercial options available or you can sign your own if you have the capability. The output of either process is a `server.crt` file.

10. Edit the `/blackboard_home/apps/httpd/conf/httpd.conf` file to include the following directive:

```
Include conf/ssl.conf
```

11. Edit the `/blackboard_home/config/bb-config.properties` file by modifying the following attributes, as shown below.

```
bbconfig.unix.ssl.certificatefile=/path/server.crt

bbconfig.unix.ssl.certificatekeyfile=/path/server.key
```

12. Edit the `/blackboard_home/apps/httpd/conf/ssl.conf` file to use designate the level of encryption. Blackboard recommends the use of strong encryption, for example:

```
SSL Protocol -ALL +SSLv3 +TLSv1

 SSLCipherSuite
ALL:!aNULL:!ADH:!eNULL:!LOW:!EXP:RC4+RSA:+HIGH:!MEDIUM:!SSLv2
```

13. Run PushConfigUpdate as shown below.

```
blackboard_home/tools/admin/PushConfigUpdates.sh
```

14. The SSL Choice feature can now be used to select which areas of Blackboard Learn use SSL. To learn more about using SSL Choice, see Setting Up SSL Choice.

Setting Up SSL Choice

After IIS or Apache is configured to support SSL, then the communication between users and Blackboard Learn can be configured using the SSL Choice feature. SSL Choice allows Administrators to determine if none, all, or some of Blackboard Learn is secured with SSL.

Note: If the SSL Choice is set to use SSL before SSL is configured in IIS or Apache Blackboard Learn will not be accessible! To ensure that users can always login, configure IIS or Apache for SSL prior to changing the security options on the SSL Choice page.

If planning on using SSL, Blackboard recommends enforcing SSL on the entire system. This ensures that all proprietary data is secured. If the choice option is chosen, it is important to update SSL settings whenever a new tool is enabled or a System Extension added.

How to Set Up SSL Choice

1. From the Security and Integration section of the System Control Panel, click **SSL Choice**.
2. Set the properties using the following table, and then click **Submit**.

Field	Description
System-wide	
Disable SSL System-wide	Select this option and SSL will not be used to secure any of the communication between users and Blackboard Learn.
Enable SSL System-wide	Select this option and SSL will be used to secure all of the communication between users and Blackboard Learn. Recommended.
Enable SSL for the following areas	Select this option to determine which areas of Blackboard Learn will be secured through SSL. Select the different areas from the check boxes on this page.
Specific Areas	
Select the check box for each area that should be secured using SSL.	
Tools	
Select the check box for each tool, tab, or Course content area that should be secured using SSL.	
Building Block Tools	
Select the check box for each Building Block that should be secured using SSL.	

Field	Description
Proxy Tools	
Select the check box for each Proxy Tool that should be secured using SSL.	
Web Services	
Select the check box for each Web Service that should be secured using SSL.	

Setting Up LDAP Authentication with SSL

This section explains how to configure the `authentication.properties` file settings that enable Blackboard Learn, using LDAP authentication, to communicate with an LDAP Server over SSL. No extra entries need to be added to the `authentication.properties` file. The Administrator simply needs to set the appropriate properties correctly (see table below).

Note: The SSL Choice option in the System Control Panel is used to secure communication between Blackboard Learn and the client machine. To learn more, Setting Up SSL Choice.

Property	Description
`auth.type.ldap.server_url.x`	When the LDAP server is setup to communicate over SSL, this property should be: `ldap://directory.university.edu` Administrators may need to append the port number depending upon the configuration.
`auth.type.ldap.server_ssl.x`	Must be set to "true" or "false". If set to "true" the module will attempt to connect to the LDAP directory using SSL. The LDAP server must be set up to handle SSL connections.

Run `PushConfigUpdates` after editing the properties file. To learn more, see Appendix A: PushConfigUpdates. Finally, copy the JSSE, JNET, and JCERT files from `apps/tomcat/shared/lib` and paste these files into the `$JAVA_HOME/jre/lib/ext directory`.

How to Configure LDAP Authentication with SSL for the JAVA Runtime Environment (JRE)

1. Copy the following three files to the `JAVA_HOME\jre\lib\ext` directory:

```
\blackboard\systemlib\jcert-1.0.2.jar
\blackboard\systemlib\jnet-1.0.2.jar
\blackboard\systemlib\jsse-1.0.2.jar
```

2. Add the following to the `JAVA_HOME/jre/lib/security/java.security` file:

```
security.provider.1=sun.security.provider.Sun
security.provider.2=com.sun.net.ssl.internal.ssl.Provider
```

3. If there are already security providers listed, and the first one is `sun.security.provider.Sun`, a `security.provider.X` entry should be added to the end of the list.

4. Import the signed public SSL certificate. Administrators configuring a fresh install of Blackboard Learn should import a certificate for each LDAP server to the applications server's repository of trusted certificates. This is done through the keytool utility.

   Windows: http://java.sun.com/j2se/1.3/docs/tooldocs/win32/keytool.html

   UNIX: http://java.sun.com/j2se/1.3/docs/tooldocs/solaris/keytool.html

How to Configure Contextual Error Messages for LDAP

The default error message does not report the cause of the error to the user when LDAP Authentication fails. Use the `auth.type.ldap.context_factory` property to display contextual error messages to users.

For each LDAP server configured, set the `auth.type.ldap.context_factory` property. This property references a class to use for creating contexts, which need to be server specific. The options are:

`blackboard.platform.security.authentication.ResponseControlContextFactor`

Any server that supports the Netscape password policy response controls spec - any breed of Netscape Directory Server including RedHat, SunONE, OpenLDAP, and others.

`blackboard.platform.security.authentication.PasswordPolicyContextFactory`

Any server that supports the IETF standard password policy attributes `(passwordExpirationTime)` but not response controls, for example, Novell Directory Server.

The `PasswordPolicyContextFactory` is used by default. If the `PasswordPolicyContextFactory` is used the authentication module must be configured to use a valid privileged user (one that can search and access the `passwordExpirationTime` attribute of any user) for the proper error message to be shown.

Setting Up SSL for SIF

The connection between the ZIS and the SIF Agent can be secured using SSL. The settings are easily configured in the `bb-config.properties`, however, SSL encryption requires a keystore and possibly a truststore from the ZIS server to function properly. The instructions below explain how to create and configure a keystore as well as how to configure a truststore from the ZIS server.

To learn more about using SSL with SIF integration, review the SifWorks ADK documentation available from Edustructures.

Creating and Configuring the Keystore

How to Create and Configure the Keystore on Windows

1. Run the following from the command line:

   ```
   %JAVA_HOME%\bin\keytool -genkey -v -keystore
   C:\blackboard_home\config\certs\SIFagent.ks -alias SIFagent -keyalg RSA -
   keysize 1024
   ```

2. The first prompt asks for a password for the keystore. The default password is "changeit".
3. The next few prompts ask for information about the person creating the certificate. This information will appear to users when they first access the Collaboration Tool over SSL. Users are prompted to accept the certificate so it is important to provide accurate information so that users trust the certificate. The information recorded is:
   - First and Last Name
   - Organizational Unit
   - Organization
   - City or Locality
   - State or Province
   - Two-letter country code
4. The last prompt asks for the password for the certificate. This password must be the same as the password entered in Step 2. Press ENTER to confirm that the same password will be used.
5. The keystore will be created in the specified directory.
6. Create a Certificate Signing Request (CSR) for the SIF agent and sign the certificate.

```
%JAVA_HOME%\bin\keytool -certreq -keystore
C:\blackboard_home\config\certs\SIFagent.ks -alias SIFagent -file
SIFagent.csr
```

7. Submit the CSR to a certifying authority (CA) or self-sign the certificate.
8. Download the server certificate and the CA certificate and copy them to the ZIS server.
9. Import the server certificate into the Blackboard server keystore.

```
%JAVA_HOME%\bin\keytool -import -alias NAMEcaroot -file
C:\blackboard_home\config\certs\NAME.cer -keystore
C:\blackboard_home\config\certs\SIFagent.ks

keytool -import -alias SIFagent -file
C:\blackboard_home\config\certs\SIFagent.cer -keystore
C:\blackboard_home\config\certs\SIFagent.ks

keytool -list -keystore C:\blackboard_home\config\certs\SIFagent.ks -
storepass changeit
```

10. Share the keystore with the ZIS server.

How to Create and Configure the Keystore on UNIX

1. Create a directory within the blackboard directory to hold the certificate.

```
mkdir /blackboard_home/config/certs

cd /blackboard_home/config/certs
```

2. Create a keystore by running the following command and responding to the prompts:

```
keytool -genkey -v -keystore SIFagent.ks -alias SIFagent -keyalg RSA -
keysize 1024
```

   - Enter keystore password: changeit
   - What is your first and last name? first last
   - What is the name of your organizational unit? Product Development
   - What is the name of your organization? Blackboard Inc
   - What is the name of your city or locality? Washington
   - What is the name of your state or province? DC
   - What is the two letter country code for this unit? US
   - Is CN=first last, OU=Product Development, O=Blackboard Inc, L=Washington, ST=DC, C=US correct? Yes
   - Enter key password for (RETURN if same as keystore password): RETURN

3. Create a Certificate Signing Request (CSR) for the SIF agent and sign the certificate.

```
keytool -certreq -keystore SIFagent.ks -alias SIFagent -file
SIFagent.csr
```

4. Submit the CSR to a certifying authority (CA) or self-sign the certificate.
5. Download the server certificate and the CA certificate and copy them to the ZIS server.
6. Import the server certificate into the Blackboard server keystore.

```
cd /blackboard_home/config/certs
keytool -import -alias NAMEcaroot -file NAME.cer -keystore SIFagent.ks

keytool -import -alias SIFagent -file SIFagent.cer -keystore SIFagent.ks

keytool -list -keystore SIFagent.ks -storepass changeit
```

7. Share the keystore with the ZIS server.

Configuring TrustStore

How to Configure TrustStore on Windows

Import the ZIS server certificate into the SIF Agent Trusted keystore.

```
cd C:\blackboard_home\config\certs

%JAVA_HOME%\bin\keytool -import -v -alias SIFWorks -keystore Trusted.ks -file
ZIS.cer

Trust this certificate? [no]: yes

%JAVA_HOME%\bin\keytool -list -keystore Trusted.ks -storepass changeit
```

This will create a new keystore containing the ZIS certificate, which will be trusted.

How to Configure TrustStore on UNIX

Import the ZIS server certificate into the SIF Agent Trusted keystore.

```
cd /blackboard_home/config/certs

keytool -import -v -alias SIFWorks -keystore Trusted.ks -file ZIS.cer

Trust this certificate? [no]: yes

keytool -list -keystore Trusted.ks -storepass changeit
```

This will create a new keystore containing the ZIS certificate, which will be trusted.

Setting Up SSL for the Collaboration Tool in Windows

Setting up SSL to encrypt connections to Blackboard Learn does not secure the Collaboration Tool because the Collaboration Tool uses Tomcat, not Apache or IIS, to handle user connections and serve pages. Securing the Collaboration Tool requires using a separate SSL certificate with Tomcat.

Most Institutions do not need to worry about securing the Collaboration Tool because the Collaboration Tool is not used to transmit sensitive data. It should also be noted that using SSL with the Collaboration Tool slows down performance of the tool. Consider both the need for security and the performance slow down associated with applying SSL before deciding to use SSL with the Collaboration Tool.

As part of the process, a keystore and a self-signed certificate are created. A keystore is a file that stores certificates. A self-signed certificate is a certificate created by you that is not submitted to a Certifying Authority.

Note: Macintosh users running Netscape, Internet Explorer, or Safari may use self-signed certificates to configure SSL. A pop-up warning may appear during the process; select **Continue** to complete the process.

If users would prefer to use a signed certificate see the Java documentation on keytools for information on obtaining a signed certificate and including it in the keystore.

In most cases, taking the extra step to go through a Certifying Authority is not necessary when securing the Collaboration Tool. Certifying Authorities are used to prove to users of a Web site that the connection is secure and verified by a trusted third party. Users accessing the Collaboration Tool from your Blackboard Learn most likely do not require the validation of a third party before using the tool.

How to Configure the Collaboration Tool with a Self-signed Certificate

1. Create a keystore.
2. Configure Tomcat properties to use SSL encryption.

Load-Balanced Configurations

The same certificate must be used on each server. For detailed instructions on how to install the same certificate on each server, see Microsoft Knowledge Base article 310178 at http://support.microsoft.com/default.aspx?scid=kb;en-us;310178&Product-win2000.

Services on each Web/application server must be restarted after changing the SSL Choice option.

How to Create the Keystore

After creation the keystore contains a self-signed SSL certificate specifically for Tomcat, `<tomcat>`. To create the keystore and certificate, follow these steps:

1. Log on to the Web/app server as the user that runs Blackboard Learn.
2. Run the following from the command line:

   ```
   %JAVA_HOME%\bin\keytool -genkey -storetype pkcs12 -alias tomcat -keyalg
   RSA -keystore <path_to_keystore>
   ```

   The keystore will be created at the `<path_to_keystore>`.
3. The first prompt asks for a password for the keystore. The default password that Tomcat expects is "changeit," but it is recommended that another password be used. Tomcat can be configured later to accept the new password.
4. The next few prompts ask for information about the person creating the certificate. This information will appear to users when they first access the Collaboration Tool over SSL. Users are prompted to accept the certificate so it is important to provide accurate information so that users trust the certificate. The information recorded is:
   - First and Last Name
   - Organizational Unit
   - Organization
   - City or Locality
   - State or Province
   - Two-letter country code
5. The last prompt asks for the password for the `<tomcat>` certificate. This password must be the same as the password entered in Step 2. Simply press `ENTER` to confirm that the same password will be used.

   The keystore will be created in the specified directory.

How to Configure Tomcat to Work with the SSL Certificate

After creating the keystore and certificate, the last step is to edit the `blackboard\config\bb-config.properties` file. Follow these steps to edit the file to work with SSL:

1. Make a backup of the following file:

   ```
   blackboard\config\bb-config.properties
   ```
2. Keep it safe so that the original settings can be restored.
3. Open the `bb-config.properties` file in Notepad or an XML editor.

4. Find the following lines in the file and add the appropriate values.

```
bbconfig.collabserver.keystore.filename=
bbconfig.collabserver.keystore.password=
bbconfig.collabserver.portnumber.ssl.default=8443
bbconfig.collabserver.keystore.type=PKCS12
```

   The `keystore.type` must be set to `PKCS12`.

5. Save the file.
6. Run `PushConfigUpdates` to apply the changes. To learn more, see Appendix A: PushConfigUpdates.
7. Test the system. When accessing the Collaboration Tool, a prompt should appear to accept the certificate. After accepting the certificate, the Collaboration Tool will open and communications will be secured using SSL encryption.

Setting Up SSL for the Collaboration Tool in UNIX

Setting up SSL to encrypt connections to Blackboard Learn does not secure the Collaboration Tool because the Collaboration Tool uses Tomcat, not Apache or IIS, to handle user connections and serve pages. Securing the Collaboration Tool requires using a separate SSL certificate with Tomcat.

Most Institutions do not need to worry about securing the Collaboration Tool because the Collaboration Tool is not used to transmit sensitive data. It should also be noted that using SSL with the Collaboration Tool slows down performance of the tool. Consider both the need for security and the performance slow down associated with applying SSL before deciding to use SSL with the Collaboration Tool.

As part of the process, a keystore and a self-signed certificate are created. A keystore is a file that stores certificates. A self-signed certificate is a certificate created by you that is not submitted to a Certifying Authority.

Macintosh users running a Netscape or Internet Explorer browser will not be able to access the Collaboration Tool if a self-signed certificate is used to configure SSL. The Safari Web browser will work with a self-signed certificate.

If there are Macintosh users running Netscape or Internet Explorer browsers then use a signed certificate. If a signed certificate is preferred, see the Java documentation on keytools for information on obtaining a signed certificate and including it in the keystore.

In most cases, taking the extra step to go through a Certifying Authority is not necessary when securing the Collaboration Tool and a self-signed certificate may be used. Certifying Authorities are used to prove to users of a Web site that the connection is secure and verified by a trusted third party. Users accessing the Collaboration Tool from your Blackboard Learn most likely do not require the validation of a third party before using the tool.

How to Configure the Collaboration Tool with a Self-signed Certificate

1. Create a keystore.
2. Configure Tomcat properties to use SSL encryption.

How to Configure the Collaboration Tool with a Signed Certificate

Clients who would like to use their existing SSL certificate should follow these steps.

1. Convert the `server.key` and `server.crt` into a PKCS12 keystore using OpenSSL.

```
openssl pkcs12 -export -out keystore.pkcs12 -in /path/to/server.crt
-inkey /path/to/server.key
```

2. This will prompt for a keystore password. The keystore will be created as `keystore.pkcs12` in the current directory. Move this to an appropriate location.

3. Use the keystore and certificate in the steps below that cover editing the `bb-config.properties` file so that Tomcat uses SSL.

How to Create the Keystore

After creation, the keystore contains a self-signed SSL certificate specifically for Tomcat, `<tomcat>`.

To create the keystore and certificate, follow these steps:

1. Log on to the Web/app server as the user that runs Blackboard Learn.
2. Run the following from the command line:

```
%JAVA_HOME%\bin\keytool -genkey -storetype pkcs12 -alias tomcat -keyalg
RSA -keystore path_to_keystore
```

   The keystore will be created at the *path_to_keystore*.

3. The first prompt asks for a password for the keystore. The default password that Tomcat expects is "changeit", but it is recommended that another password be used. Tomcat can be configured later to accept the new password.
4. The next few prompts ask for information about the person creating the certificate. This information will appear to users when they first access the Collaboration Tool over SSL. Users are prompted to accept the certificate so it is important to provide accurate information so that users trust the certificate. The information recorded is:
   - First and Last Name
   - Organizational Unit
   - Organization
   - City or Locality
   - State or Province
   - Two-letter country code
5. The last prompt asks for the password for the `<tomcat>` certificate. This password must be the same as the password entered in Step 2. Simply press `ENTER` to confirm that the same password will be used.
6. The keystore will be created in the specified directory.

How to Configure Tomcat to Work with the SSL Certificate

After creating the keystore and certificate, the last step is to edit the `/blackboard/config/bb-config.properties` file. Follow these steps to edit the file to work with SSL:

1. Make a backup of the `/blackboard/config/bb-config.properties` file.
2. Keep it safe so that the original settings can be restored.
3. Open the `bb-config.properties` file in an editor.
4. Find the following lines in the file and add the appropriate values.

```
bbconfig.collabserver.keystore.filename=
bbconfig.collabserver.keystore.password=
bbconfig.collabserver.portnumber.ssl.default=8443
bbconfig.collabserver.keystore.type=PKCS12
```

   The `keystore.type` must be set to `PKCS12`

5. Save the file.
6. Run `PushConfigUpdates` to apply the changes. To learn more, see Appendix A: PushConfigUpdates.
7. Test the system. When accessing the Collaboration Tool, a prompt should appear to accept the certificate. After accepting the certificate, the Collaboration Tool will open and communications will be secured using SSL encryption.

Appendix A: PushConfigUpdates

This tool updates the configuration according to the settings in the `bb.config.properties` file.

WARNING! Running this command will redeploy all of the properties files. If any customizations have been made to the properties files, they will be lost.

The `PushConfigUpdates` command has been enhanced to improve system management. Now, the `PushConfigUpdates` automatically updates the admin data in the database by reading the value in the `config.xml`. It automatically pushes the changes of the database hostname and port, instance name, and externally visible Web server hostname to the database. Running this tool always restarts the services to reflect the changes.

The first operation of this tool will replace the existing template files, copying the original template files to a time-stamped sub-directory of `blackboard_home/backups/templates/time_stamped`. Use these files to retrieve and re-apply any local customizations.

The second operation of this tool is Tomcat specific and requires that Custom Authentication be disabled to successfully complete this operation. The .jar files from `apps/tomcat/server/lib/directories` will be loaded rather than from `blackboard_home/systemlib/`. Be aware that any .jar file found in the directory will be loaded at Tomcat startup. This operation is controlled by the .classpath files located in `config/tomcat/classpath`. Any changes to the Tomcat configuration files or startup scripts must be made to the templates in the `config/tomcat/` directory, in particular this applies to additional MIME types added to the web.xml file. Touch points are files such as web.xml, server.xml, startup scripts, and configuration files used in clustered Tomcat environments.

WARNING! Tomcat Clustering is deprecated for Blackboard Learn 9.1 Service Pack 6, and will be removed from Blackboard Learn 9.1 Service Pack 8.

The third operation updates the BBLEARN.SYSTEM_REGISTRY (legacy: BB_BB60.SYSTEM_REGISTRY) database table with the configuration changes. The current performance parameters for the Application server are recorded in the BBLEARN_ADMIN.CONFIG.REGISTRY (legacy: BBADMIN.CONFIG.REGISTRY) database table.

The final operation configures content management, which includes license verification, connection information update, then pushing the new information to the database. The version of each database schema is then checked and updated if necessary.

Windows:

```
blackboard_home\tools\admin\PushConfigUpdates.bat
```

UNIX:

```
blackboard_home/tools/admin/PushConfigUpdates.sh
```

When using the `PushConfigUpdates` tool in Windows, it is very important that the tool is run on the command line rather than double-clicking the file from windows explorer. The command line will execute the tool in verbose mode, displaying important messages.