Setting up an Institution to Use WASE:

The following steps detail what must be done to set up an institution to run WASE. In general, you carry out these steps either from a web browser or while logged in to the production WASE server, either as user 'wasp' or as 'root' (details below). Many of the steps involve manipulating a yml (YAML) file. BE CAREFUL: yml files have a very strict format … if you get them wrong, WASE will die with no user-visible message … there will be an error message in the WASE error log, which is located on the production WASE server at:

/usr/local/wasp/php-fpm/logs/error.log

You may want to open a shell window and do a tail –f on this file as you go through the following steps (except the first one).

Step 1: Determine if the institution can use WASE

You will want to have the institution first determine if WASE will be useful to them. The basic documentation is at:

https:/wase.princeton.edu/princeton/docs/WASE.docx or
https:/wase.princeton.edu/princeton/docs/WASE.pdf

If the institution wants to run WASE, then you must determine if they can run WASE. There are two key requirements: LDAP)directory access, and authentication.

First, authentication. WASE has to be able to authenticate the institution users. WASE supports 4 kinds of authentication:

SHIBBOLETH: most widely applicable. Very easy if institution is in the InCommon federation (but not a requirement).

CAS: Very easy to set up, but their CAS server has to accept connections from WASE.

LDAP: Very easy to set up, but their LDAP server has to accept connections from WASE. LDAP is also used for directory lookups, if enabled.

YOU MUST OPEN A FIREWALL PORT AT PRINCETON TO ENABLE OUTGOING LDAP CONNECTIONS.

The remote LDAP server may use ldaps (pot 636) for connections, in which case it will have a certificate (CA cert). The openldap code that PHP uses must be able to validate that cert, so you may need to get a copy of this cert (it will be in a file with an extension of "pem") and copy it to the /etc/openldap/ccacert directory. You may also need to update the ldap.conf file in /etc/openldap. You MUST restart the php-fpm service once you have made this update.

LOCAL: Userid/password stored locally. This is really a skeletal capability, and requires use of phpMyAdmin to manage the local authentication table (waseUser). **Very useful when you want to allow a remote institution to evaluate WASE.** There is a local "test" institution site that you can point people to.

Next, LDAP.   Although it is possible to run WASE without LDAP access, a lot of things will not work. For example, users cannot add managers or members to their calendars, because those require LDAP lookups to get email addresses.

If the institution supports one of the above, and LDAP access, continue. Else tell them you're sorry.

Step 2: Name the institution

Decide on a short name for the institution. Something like "princeton", "cornell", whatever. This short name must not conflict with any existing short names. For the rest of this document, we will use "academe" as the short name of the target institution. **<u>Substitute the real target short name for "academe" in all of the steps below.</u>**

Step 3: Create and initialize the institution config directory

Login as user 'wasp' to the production WASE server, wspprod03l. The server is behind a bastion host (epoxy), so you will need your fob to do this. You must be root to do the following, So the next step is to become root. To do this:

Su to a user who can su root (e.g., 'su serge').
Once you are running as the above use (e.g., 'serge'), do 'sudo su root'. This should cause you to get a fob prompt. Use your fob to enter the passcode. You should now be root.

Cd to the production 'config' diurectory: cd `/usr/local/wasp/www/wasp/config`

Create an "academe" directory under the top-level config directory. Give this directory 775 permissions and assign it to user 'wasp' and group 'nginx':

From within the config/academe directory: chmod 775 . ; chgrp nginx . ; chown wasp .

Make sure you get the ownership/permissions right, or else a lot of things will fail.

Copy the templates/config/waseParmsCustom.yml and waseParmsSystem.yml files into this directory.   These files will be edited to become the institutional parameters files. Make sure that the permissions of these files is 660, the owner is wasp and the group is nginx. You can now su back to user wasp.

Step 4: Create the MySQL database for the institution

You do this from a web browser.

1) Point your browser to waspprod03l.princeton.edu/phpMyAdmin and login as user "root", password "7ei…".
2) In the "Create new database" prompt, enter "waseacademe" and set collation to "utf8_general_ci". Create the database.
3) Click on the "home" icon at top left of the phpMyAdmin panel, then click on "privileges" in center of the screen.
4) Click "Add a new User"
5) Enter "waseacademe" in the User name field. Set the host to "localhost". Enter a password (**write it down**!) in the "Use text field" box. Repeat the password in the Re-type field. Click Go to add the user (do not specify any global privileges).
6) Select Add privileges on the following database, and select "waseacademe" from the drop-down list (or enter the database name in the text field).
7) Select Check All, and click Go. Make sure you have given waseacademe all privs for the waseacademe database.

Note: the database password created above becomes the administrator password for the institution. It will be referred to as the "mysqlpassword" in what follows.

Step 5: Start filling in the waseParmsSystem.yml file

From the WASE production server, logged in as user 'wasp', fill in the database name ("waseacademe"), the MySQL user name ("waseacademe"), and the mysqlpassword in the academe/waseParmsSystem.yml file (parameters DATABASE:, USER: and PASS:). You must also fill in the (short) institution name, and as many of the other parameters as you can.

Step 6: Create the MySQL tables for the institution

Run the maketables script from a web browser as follows:

https://wase.princeton.edu/academe/admin/maketables.php?secret=mysqlpassword

You should get a report back saying that all of the tables were created. If not, fix any problems encountered and repeat (it is ok to re-run the maketables script, even if some tables were created already). If you get a permissions error, then you did not set the privs correctly in step 4. Go back and fix them.

Step 7: Set up authentication.

From the WASE production server, logged in as user 'wasp', fill in the authentication parameters in the waseParmsSystem.yml file in the institution's config directory. You will need to work with the institution's identity management folks to get this set up. There is a write up below on each of the possible authentication methods and how to set them up, so read and follow the directions for the selected authentication method, then come back here to continue.

Step 8: Set up integrations

WASE can work with a number of external systems. These currently include:

Exchange: for calendar synchronization
Google: for calendar synchronization
LTI: for access from the institution LMS

These integrations do NOT need to be done before an institution can start testing WASE, and do not need to be done at all. They are all optional.

Setting up Google:

WASE comes pre-configured to read/write Google calendars. When an institution user selects Google as their sync calendar in WASE Preferences, they are taken through an OAUTH2 authorization procedure whereby they allow WASE to update their Google calendars. The configuration information for Google integration is in the waseParmsSystem.yml file, and does not vary across institutions. However, you must add the institution authorizegoogle URL to the list of authorized Google redirect URLs, as follows:

Login to Google as user wasp2gcal@gmail.com
The password is the 7ei database password.
Go to the Google developers console:
https://console.developers.google.com
Click on "Credentials"
Click on Web Client 1. You should see a list of Authorized redirect URIs.
Type in the authorizegoogle URI:
https://wase.princeton.edu/academe/views/pages/authorizegoogle.php

Click Save.

Setting up Exchange:

You can configure an institution to have a user's appointments/blocks synced into their exchange calendars. The way this works is as follows:

1) The institution designates an Exchange userid/email/password to which users will give write access to their calendar. This is an exchange user on the institution's Exchange server.
2) The institution transmits that user/emai/password to you. From the WASE production server, logged in as user 'WASE', fill in the Exchange parameters in the WASEParmsSystem.yml file as follows:

```
# Exchange server host name
EXCHANGE_HOST: "something.exchange.academe.edu"
# The email address for login to Exchange
EXCHANGE_EMAIL: "WASEuser@exchange.academe.edu"
# The username for login to Exchange
EXCHANGE_USER: "WASEuser"
# The password for login to Exchange
EXCHANGE_PASSWORD: "something"
# The type of Exchange integration:  direct or not (by invitation)
EXCHANGE_DIRECT: 1
```

3) Institution users must give the above Exchange user write access to their Exchange calendars. WASE explains how to do this when the user selects Exchange as their sync preference in the WASE Preferences system.

Setting up LTI:

This is a covered in the LTI section below.

Step 9: Set the institutional parms (WASEParmsCustom.yml)

Have the institutional representative fill in the WASEParmsCustom.yml file using the “parms.php” script, as follows:

https://WASE.princeton.edu/academe/admin/parms.php?secret=mysqlpassword

Note that you have to substitute in two places in the above URL: academe and mysqlpassword.

You may well have to help the institution fill in this file. To do this, login to the WASE production server as user ‘WASE’, and edit the institution’s WASEParmsCustom.yml file. Most fields can be left in their default state.

Step 10: Validate the institutional parms

You will need to carefully review the parms set by the institution, particularly in the authentication section.

Make sure that the institution has not specified an LDAP server unless that server is accessible to WASE; otherwise, WASE will hang.

You may want to set up the institution’s academic calendar for them. To do that, find their online academic calendars, figure out day types, then run the admin/calendar.php script for them to assign days to daytypes.

Step 11: Set up reminders and backups

You need to set a cron job to run the reminder script for the institution, and insure that database backups are taking place.

Make sure you are logged in to the WASE production server as user 'WASE'. Do a 'crontab –e' to edit the crontab database. Yank and put a copy of one of the other institution reminder lines and changed the institution name to match the new institution. These cron job run the reminder script at 5pm local time; you may want to reset that to correspond to 5pm in the target institution's timezone.

You should not need to do anything to add the new database to the backup schedule. WASE runs the 'automysqlback' script, out of /usr/local/WASE/www/automysqlbackup. The backups are triggered out of cron. There is an automysqlbackup.conf file in this directory which controls backups. That file is set up to backup all of the mysql databases, so it should pick up your new database without you're having to do anything. After a full day has passed (backups run at night) check the `/usr/local/WASE/www/automysqlbackup/automysqlbackups/daily directory … you should see an entry for your new database.`

Step 12: Admin documentation

Now that the installation is set up, you can point the installation administrator to the documentation. The basic documentation is at:

https://WASE.princeton.edu/academe/docs/WASE.docx or
https://WASE.princeton.edu/academe/docs/WASE.pdf

The above documents can be accessed without a login. The administration documents require a login. They are at:

https://WASE.princeton.edu/academe/admin/admindoc.php or
https://WASE.princeton.edu/academe/admin/adminpdf.php

AUTHENTICATION:

SHIBBOLETH:

We are running a SHIB SP (service provider). The URL is:

https://WASE.princeton.edu/simplesaml

The code is on the WASE production server, at:

/usr/local/WASE/www/simplesamlphp

To access the system as an administrator, point a web browser at:

https://WASE.princeton.edu/simplesaml

Click on the "Configuration" tab, then on "Login as administrator". Leave the username as admin. The password is the short "7ei…".

To add a non-InCommon institution for SHIB authentication:

There are a bunch of tabs on the simplesaml screen. Click on the Federation tab. You will see a screen that lists the SHIB 2.0 SP Metadata. Click on the Show metadata link. This will display a page of meta-data. Cut-and-paste that metadata and email it to the institution's SHIB administrator. You can also let this administrator know that the WASE entity ID is:

https://WASE.princeton.edu/simplesaml/module.php/saml/sp/metadata.php/default-sp

To add an InCommon institution for SHIB authentication:

You don't need to send anything to the institution's SHIB administrator: the WASE metadata is part of the InCommon metadata feed.

For both InCommon and Non-InCommon institutions

You need to add the institution's SHIB mete-data to the simplesaml configuration files (the shib13-idp-remote.php and the saml20-idp-remote.php files). The institution SHIB administrator will need to send you, or point you to, their SHIB IdP metadata (this is usually public). The following instructions are taken from the document:

https://simplesamlphp.org/docs/stable/simplesamlphp-sp

> The service provider you are configuring needs to know about the identity providers you are going to connect to it. This is configured by metadata stored in `metadata/saml20-idp-remote.php` and `metadata/shib13-idp-remote.php`. This is a minimal example of m`etadata/saml20-idp-remote.php` metadata file:

```
<?php
$metadata['https://openidp.feide.no'] = array(
    'SingleSignOnService'  =>
'https://openidp.feide.no/simplesaml/saml2/idp/SSOService.php',
    'SingleLogoutService'  =>
'https://openidp.feide.no/simplesaml/saml2/idp/SingleLogoutService.php'
,
    'certFingerprint'      =>
'c9ed4dfb07caf13fc21e0fec1572047eb8a7a4cb',
);
```

For more information about available options in the idp-remote metadata files, see the IdP remote reference.

If you have the metadata of the remote IdP as an XML file, you can use the built-in XML to simpleSAMLphp metadata converter, which by default is available as `/admin/metadata-converter.php` in your simpleSAMLphp installation.

Note that the idp-remote file lists all IdPs you trust. You should remove all IdPs that you don't use.

You should test the SHIB setup by going to the Authentication tab in simplesaml and clicking on "test configured authentication sources". Select your new IdP, and see if you can get to it.

You need to have the institution SHIB administrator release the userid (cn) attribute to WASE (that is required), as well as as-many of the following as they can give you:

SHIBUSERID:, SHIBNAME:, SHIBOFFICE:, SHIBEMAIL: and SHIBPHONE:

The critical step is to get the names they are using for these attributes. For example, with InCommon, the userid attribute is usually called 'cn'. Some institutions, however, pass attribute names that begin with 'urn:'. Whatever they tell you will probably be wrong … set the 'DUMPSHIBATTRIBUTES' parm in WASEParmsSystem.yml to your email address to have WASE email you the attribute array it gets back from their SHIB server at login, and then set the attribute name parameters to the correct values in their WASEParmsSystem.yml file.

Set the institution entityID value in the SHIBIDP parm of the WASEParmsSystem.yml file. You also need to set the attribute names for the SHIBUSERID:, SHIBNAME:, SHIBOFFICE:, SHIBEMAIL: and SHIBPHONE: attributes. SHIBUSERID is the most critical; if you get that wrong, nothing will work. It is typically 'cn' or 'uid' or something that starts with 'urn:'.

CAS:

If academe wants to use CAS to authenticate, they need to also allow WASE to access their CAS server. Get their CAS protocol version, host, port, login and logout URL, and URI, and update their WASEParmsSystem.yml file with the appropriate values. Make sure you specify the correct (and valid) CAS version, host, port and URI. It is NOT enough for you to be able to access their CAS server via a browser --- all of the CAS parms must be properly set, or you will get a failure in the phpcas library code.

MAKE SURE THE CAS PROTOCOL VERSION IS CORRECT. It is probably “2.0”, not “3.0”. Even if the target school is running a “3.x” release of CAS.

Some CAS servers are set up to return attributes. If so, determine if they can pass back any of the SHIB attributes listed in the WASEParmsSystem.yml file (WASE uses the same parameter names for CAS and SHIB), and then specify the attribute names passed back in SHIBOFFICE, SHIBNAME, SHIBEMAIL and SHIBPHONE. If their LDAP server accepts remote connections, fill in the LDAP configuration parameters (but do NOT fill these in if they have a firewall blocking external access, as this will cause WASE to hang).

As with SHIB, you can set the ‘DUMPSHIBATTRIBUTES’ parm to your email address to have WASE email you the attribute array it gets back from their CAS server to make sure you specified them correctly.

LDAP:

WASE can use LDAP for authentication and/or directory lookups. In either case, the institution LDAP server must be accessible to WASE (not behind a firewall). Make sure this is the case before configuring LDAP in the WASEParmsSystem.yml file.

YOU MUST ALSO OPEN A FIREWALL PORT AT PRINCETON TO ENABLE OUTGOING LDAP CONNECTIONS.

YOU SHOULD ENCODE THE LDAP HOST NAME AS: “ldaps://server.institution.edu:port”, rather than just giving a host name (or try both to see which one works).

You need to fill in the various LDAP parameters. If the institution LDAP server does not support anonymous binds, then you need to fill in an LDAP login userid/password. Make sure you use the right LDAP attribute names for the various fields that can be returned by LDAP.

LOCAL:

WASE will happily validate userids/passwords from a local database (WASEUser). The values in this database need to be filled in using phpMyAdmin. I have configured a “test” institution site that uses a local database that you can point users to try WASE. It has the following user/passwords:

Prof1/Prof1
Prof2/Prof2
Stud1/Stud1

Stud2/Stud2
Serge/secret

Serge is configured as a super_user. The first four netids/passwords are advertised on the didyouknow panel of the login screen.

LTI:

WASE can function as an LTI 1.0+ tool provider. The 1.0 protocol supports shared authentication. The 1.2 protocol additionally provides for roster access, which allows WASE to take a student directly to a list of their instructor's WASE calendars (if they have any). WASE will test to see if roster information is available, and use it if it is.

Three parameters control the LTI integration on the WASE end: LTILAUNCH, LTIKEY and LTISECRET. The first points to the LTI code in WASE, and can be set based purely on the short name for the institution. LTIKEY and LTISECRET correspond to the LTI shared "key" and shared "secret"; all three parameters must be set identically in WASE and in the institution's LMS. Additional information on configuring LTI is available in the WASEAdmin.doc file.

NOTE: When LTI is configured, it is very important the institution correctly configure the LMS element that will be passed to WASE as the userid field. It must match the login userid in WASE.

phpMyAdmin:

You can access the WASE databases either by logging in to WASEprod03l, or via phpMyAdmin, as follows:

https://WASEprod03l.princeton.edu/phpMyAdmin

Login as "root" using the "7ei…" password (short, database form). Note that you must use "WASEprod03l" in the above URL, not "WASE". Be careful mucking with the databases, as there are many dependencies. The safest thing to do is to only do SELECTs.