Intranet 2 Manual

Bryan Jacobs (with contributions from others)

Last updated February 27, 2007

Contents

Chapter 1

Intranet 2 for Users

1.1 What is Intranet 2?

1.1.1 What is it?

Intranet 2 the second version of a web site which makes life easier for TJ students, teachers, and administrators. It is a central component of the eighth-period system. Being able to use the Intranet is a valuable skill for any TJ student.

1.1.2 Why was it developed?

The original Intranet was developed in the age of the dinosaurs (well actually, more like the late '90s) by TJ students. It was written to work, plain and simple. It did work rather well and quickly became an important part of TJ life. However, when the original Intranet was written, attention was paid only to getting it to function: the resulting system was very difficult to modify. None of the current TJ students understood its black magic, and it was impossible to add any cool new features. Something had to be done.

A group of enterprising TJ students assembled in the Computer Systems Lab. Their mission was clear: improve the Intranet. They quickly determined that it would be difficult (if not downright impossible) to fight the spaghettilike morass of the current Intranet, rife with statements such as:

```
if (is_sysadmin());
```

The technically inclined reader would note that this code does not, in fact, *do* anything. However, the technically inclined reader would, in fact, be decieved (don't worry if this is you – you're in good company). This code exists so that if the condition is true, neither the following *else if* nor *else* blocks are executed. (If you are crying right now, again, you are in good company). But this is a direct excerpt from the original Intranet code. Even worse were sections such as the following, the body of the is_sysadmin method:

```
if ($user=='dtran' || $user=='edanaher' || $user=='agupta') {
        $showstuff = true;
} else {
        $showstuff = false;
}
```

So, the Intranet 2 team decided that they would rewrite the Intranet from the ground up, designing it from the very beginning to be easy to modify and extend, with clear documentation and sensible coding practices.

1.1.3 What can I do with it?

With Intranet 2, you can:

- Check your email
- Sign up for Eighth period
- Look up contact information about TJ students or teachers
- Look at class schedules
- Vote in polls and elections
- Read school news and announcements
- Sign up for a parking space (Juniors and Seniors only)
- Sign up for the AMC
- Anything else that some student decides to make possible!

1.2 Logging in

To log in to Intranet 2, point your web browser to https://iodine.tjhsst.edu. The site may also be accessed via intranet.tjhsst.edu.

1.2.1 The beginning of the year

At the very start of each school year, all students' passwords are reset. You may not log in to the Intranet until you have changed your password through one of the school computers. This is to make sure that nobody can log in to the Intranet using your account before you change your password. Note that this means *YOU CAN NEVER RESET YOUR PASSWORD TO YOUR STUDENT ID NUMBER*. If you do, you will be unable to log in to the Intranet.

1.2.2 Reading the news

When you log in you'll be greeted by the latest in interesting news tidbits. When you're done staring at them, click the "mark as read" button to make them go away. If you ever want to see the old news, click, not surprisingly, "old news".

1.3 Configuring Intranet

A variety of options are available by clicking on "preferences" in the top bar on the Intranet 2 main page (or at /prefs). To change your options, set the checkboxes and dropdowns the way you like and click "submit".

1.3.1 Security

There are two layers of security protections in place to hide your information: parental permissions and the checkbox-set options on the preferences page. Unless you have your parents sign and return the student information waiver you receive at the beginning of each school year, a law called FERPA (The Federal Education Right to Privacy Act) prohibits the school from displaying your information. So no matter what the settings you put on the preferences page, if your parents don't want your phone number on the Intranet it will not be.

If your parents have permitted a piece of information to appear, you still have the option to hide it by clearing the checkbox next to it in the preferences page and clicking the save button. Note that there are no options to hide IM handles or other such information which you may add to the page - if you don't want that shown, just remove it from the Intranet altogether. The hide/show options are only for information which the school will keep on file regardless of whether it is displayed on the Intranet.

Note that you may be able to see information about yourself which is not available to others, and that under certain circumstances information may be disclosed to school staff and administrators. But students searching the directory will not be able to view information you have elected not to disclose.

1.3.2 Visual appeal

The other set of options on the preferences page is Intranet 2's visual styles. Select a style and see how it looks! Choose whichever one you like and it will be saved and shown to you each time you log in.

1.4 Signing up for Eighth Period

To sign up for Eighth Period, click on the block for which you have no activity in the Eighth Period Intrabox (or visit /eighth and click on it there). Then select the activity you want and click "change".

1.4.1 What to do if you can't sign up for the activity you want

You may not be able to sign up for an activity for one or more of the following reasons:

The activity is full Most activity capacities are flexible. Try asking the Eighth Period Office to allow you to enter the activity anyway.

Eighth-period signups for today are closed (it's after lunch on the day of the activity) You need to go to the Eighth Period Office. But you really should have signed up earlier...

The activity is cancelled Tough potatoes. Find something else to do.

You have another mandatory activity If you're locked into a so-called "sticky" activity, you don't have a choice for the block. You must go to the mandatory activity.

You can't find anything you want to sign up for In the distant past, there was an "admin study hall" for students who didn't choose an activity. This no longer exists. If you do not sign up for an activity you will be treated as having skipped the eighth period block and an absence will be recorded.

You signed up for a two-block activity You can't sign up for an activity that fills both blocks and only go to one. The activity is listed as "both-blocks" because it requires all the available time.

1.4.2 Absences

Eighth-period absences are a Bad Thing (TM). You should avoid them because they hurt your chances at a good parking space, they affect your ability to sign up for certain popular activities, they make the eighth-period office mad at you, and if you get enough of them the school will call your parents. That's No Fun. If you do end up with an (undeserved!) absence, as will occasionally happen, go to the sponsor of the activity in which you are listed as being delinquent and have them sign a paper stating that you were actually present at the block or blocks under dispute. Bring said paper to the Eighth-Period Office and they will remove the absence from your record. Note that you have *30 days* from the date of the activity to resolve the absence or it will become permanent.

1.5 Checking your TJ email

To access your mail through Intranet2, you first need to go to your preferences, scroll to the bottom, add the "Your Mail" intrabox, and hit "submit". Once this is done, a preveiw of your most recent messages should appear somewhere on the left sidebar. You can then access your inbox directly (without having to retype your password!) by clicking the "Read and send messages" link.

1.6 Using the directory

1.6.1 Searching

Most of the time when you're using the directory you'll just want to find out how to get in touch with one of your classmates. To do this, search for them. The search box in the Directory module will find anyone whose first, last, middle, nick, or user name starts with or ends with what you type. If you enter multiple words the results will be people who matched ALL terms.

You may also use the "Intranet 1" style of searching. This is much more powerful than the simple Intranet2-style "type in their name" method. To invoke this method of searching, just put a colon anywhere in the search string. Again, if what you're searching for contains a colon anywhere, you'll use the old style of lookup.

Intranet 1 searches look like the following queries:

- grade:11 firstnamesound:alyssa
- first:bob last:jones
- grade<10 name:bob
- : grade>=11
- town:mclean middle:richie

The "sound" items will return anything that sounds like what you typed, regardless of spelling. Very handy, eh? Especially try out the "namesound" option.

1.6.2 Class info

A complete TJ schedule is contained within the directory. You may navigate from any student or teacher's profile to the classes they teach or take, and see the lists of students in each one (if they wish to be seen).

1.6.3 Permissions

If you want to restrict what information is available to others when they search for you, look earlier under "Configuring Intranet". Remember also that everyone else has the same options you do. If they don't want you to know their address, you can't.

1.7 Voting

Every once in a while somebody will try to establish a government. Since TJ isn't (exactly) a totalitarian regime, that process involves an election. The only fair and democratic way to hold an election is by secret ballot, with each and every member getting exactly one ballot. Except the Intranet 2 admins, who get roughly ten thousand votes apiece, with the Intranetmaster settling all disputes.

No, really, voting on Intranet 2 is a fair affair (hey, assonance!). You will be presented with a link to vote. Select the choice or choices that you think have the best-sounding names, or however else you like to vote, and click the little button at the bottom. Your vote will be recorded. You will have until the polls close to view and/or change your vote at your discretion. If you have any problems voting, contact an Intranet administrator and they will ignore you. Or help you, if you bring them cookies.

Chapter 2

Intranet 2 for Developers

2.1 Getting started with Iodine

2.1.1 Revision control

As of April 4, 2013, Intranet 2 uses the Mercurial revision-control system from http://selenic.com/mercurial. To fetch the Intranet 2 source code, run the following commands:

- mkdir intranet2
- cd intranet2
- hg init
- hg pull http://iodine.tjhsst.edu/hg
- ./setup tj

This will fetch a complete copy of the Iodine source code repository, and set it up for development. If you wish to interface with a "real" Iodine database you will need some passwords to replace those in the config.ini file. Ask a systems administrator to set you up a sandbox.

Once you are up and running, making changes to the code, here is how you can contribute:

- hg commit
- hg push

If you get a message about “creating multiple remote heads”, you have made a change against an earlier version of Iodine. Run an “hg pull” to get the remote changes, then “hg update -m” to combine the other person’s changes with yours. Finally, when your working copy looks how you want it to, “hg commit” and “hg push” to send out the final copy into the public repository.

Admins should take care to make sure that the hgrc files present in the repo and installed by the setup scripts have a correct default-push setting.

2.2 Developing Iodine

Intranet 2 is a middle-sized program—this means that unless you’ve worked on a real open-source project before it will seem mind-bogglingly complex. However, great care has been taken to make sure that the code is logically organized and broken into bite-sized yummy chunks, ready to be digested by code monkeys.

2.2.1 The application lifecycle

The following is a rough trace of the lifecycle of one page request:

1. The user requests a page using their browser.

2. The request is redirected to core.php5.

3. The module.map file is loaded and miscellaneous PHP initializers are invoked. This is the point at which the garbage collection functions are overridden.

4. core.php5 decomposes the URL into the **$I2_ARGS** variable.

5. **$I2_ERR** is initialized.

6. **$I2_LOG** is initialized.

7. **$I2_SQL** is initialized - a connection is made to the MySQL server at this point.

8. **$I2_AUTH** is initialized. Auth first looks to see if the module is ’logout’. If it is, Auth handles cleaning up anything lying around and logging out the user. Then, the Auth module checks to see if the user is logged in. If they are, we proceed. If not, the following happens:

- If the Apache REMOTE_USER variable is set, it is accepted. This means *any* Apache module may be used as Iodine authentication and Iodine will honor it. Suggested potential modules are mod_webauthldap and mod_kerberos.
- If the Iodine session variable i2_uid is set, the user is authenticated and **is_authenticated**() returns **TRUE**.
- If the user has sent credentials into the application, they are validated by the auth module specified in the config file (currently, and for the forseeable future, Kerberos). That auth module performs all necessary steps to confirm that the user identity is valid and to log them in.
- The user is not logged in, so we need to show them the login page. The background image for login is fetched by Auth (should be moved to Display, really). Then a new Display is created with the special name 'login', and it is proceeds to display the login page. The Auth constructor then calls **die**(). Execution stops with the user being shown a login page.

9. **$I2_LDAP** is initialized, and a SASL bind is made to the LDAP server (unless the user used the master password, in which case an admin bind is used).

10. **$I2_USER** is initialized. If the user is a member of the admin_ldap group, the old LDAP bind is discarded for an admin bind.

11. **$I2_DISP** is initialized. Smarty is loaded by Display. The user's style is loaded.

12. **$I2_AJAX** is initialized.

13. If the user requested a module in the URL, that module is set now—otherwise the user's default start page is used. If the module is ajax, the flow of execution halts after returning a response.

14. Display's **display_loop**(**$module**) method is invoked. If something has called **stop_display**() before this point, execution ends. Otherwise, the Nags module is given a chance to take over the page.

15. Display attempts to load the passed module. If it fails, it displays an error instead of a page.

16. A new Display instance is created for the main module by the core display.

17. The module is instantiated. If it gives an error, display catches it and recovers.

18. The module's **init_pane**() method is called. It returns the title which the page should bear, or **FALSE** if it should not be displayed to the current user at all. It may also return a two-element array—the first element is the page title, the second the title of the main pane.

19. The module's **display_pane**() method is called and passed the earlier-instantiated display object. The module displays anything which pleases it.

20. The Intrabox class' **display_boxes** method is invoked by Display. This method loops over the boxes a user wants displayed and calls **display_box** for each one. That, in turn, calls **init_box** and then **display_box** on each box module. Note that all the boxes share the core display object. However, the display object's buffering is turned on and only flushed after each box completes its output, so that unless a module misbehaves by directly manipulating the display object a module throwing an exception will just cause that module to be removed completely without disrupting the page flow.

2.2.2 Data organization and access

There are two primary data storage backends in Iodine: MySQL and LDAP. MySQL should be used for everything except contact information and information directly associated with a single user, such as their default style. LDAP provides faster read access to this data and the ability to use the school's Active Directory servers to store student information, along with the guarantee of low-level Access Control Lists guaranteeing that a coder's slip won't cause Iodine to leak sensitive information makes LDAP a solid choice for directory information. It also allows the Intranet database to be a source of a wide variety of NIS information via nss_ldap.

When creating a new module, use MySQL as its backend unless a strong justification can be made for LDAP. Place one SQL file containing a commented DROP TABLE and CREATE TABLE statement for each table your module uses into the mysql/ directory. Do not omit this step. A module then handles its own MySQL database by invoking the **query** method of the **$I2_SQL** object. The object returned may be manipulated as any PHP MySQLi-type object, via repeated **fetch_array** calls, or through a variety of well-documented convenience methods. The object *may not* be held for more than one page lifecycle. It is invalidated when the page display finishes.

All data access should occur during the init phase of the module lifecycle. The modules should locally cache the necessary information (or hold the Iodine MySQL objects and use them as a cache) so that all that needs to be done in the display phase is just that, display.

A module which uses the same information in box form as it does in pane form should not collect nor retain two copies of the same info—share!

2.2.3 Coding practices

Documentation

Document everything! The proper standard for comments is the PHPDoc format. Every file, method, and class should have a comment block. Every major variable should also have one.

Single-line comments should look like:

```
/*
** This is a single-line comment following the I2 conventions.
*/
```

or just for ease of coding's sake:

```
// This comment doesn't follow I2 conventions, but it's acceptable.
```

Do not use the C-style line comments for lengthy explanations. They should look like this:

```
/*
** This comment is quite lengthy, and so it should not be contained
** within a set of C-style //-type comments. To do so would be awkward.
** The comment should be broken into lines about every 80 characters, too.
*/
```

Important class variables, especially public ones, should be commented like so:

```
/**
 * Defines the package's foo. Currently defaults to 'bar'.
 */
 public static $foo = 'bar';
```

ALL public methods and most other important ones should be documented. The general rule is that, if anyone else could ever be reasonably expected to use the method, *or* you have to ask yourself whether you should document it, put in a PHPDoc comment like the following:

```
/**
 * The bar function calculates the bar-qwerty index of the thing
 * on which it is called.
 *
 * The bar-qwerty index is a made-up construct created just to
 * show how the first section of the comment is a short description
 * while more complicated concepts (like the bar-qwerty index) should
 * be explained later in PHPDoc's later docblock (here).  Links to other
 * {@link OtherClass classes} should be here too.
 *
 * @access public
 * @param string $bar A string representing the bar to check.
 * @param int $qwerty An integer whose value is how qwerty
 *       the function should be, defaulting to the standard
 *       qwerty value of 3.
 * @return bool Whether the function was called at all.
 */
public function foo($bar,$qwerty=3) {
        return TRUE;
}
```

And finally, at the start of each file should be a block like the following:

```
/**
* Just contains the definition for the class {@link MyClass}
* @author The Intranet 2 Development Team <intranet2@tjhsst.edu>
* @copyright 2007 The Intranet 2 Development Team
* @package MyPackage
* @subpackage ExampleSubPackage
* @filesource
*/

/**
* A demo class to show PHPDoc file and class comments
* @package MyPackage
* @subpackage ExampleSubPackage
* @see OtherRelatedClass
*/
class MyPackage {
```

The file comment must be the first line of PHP in the file. Do not omit it. The package and subpackage for the file and for the class the file contains should match.

Object orientation

Every Iodine entity should be a class or interface. All modifiers should be set using the concept of encapsulation—nothing should be made public except via public get/set methods. It is acceptable to use PHP's magical __get and __set methods to make getters and setters transparent, allowing things like the following code:

```php
        public function __get($varname) {
                if ($varname == 'meh') {
                        return $this->foo;
                }
        }

        public function bar() {
                return $this->meh+42;
        }
```

Some classes should only be instantiated once per application lifecycle. These include the data accessor classes, the Display class, and so on. Each of these classes has its own global variable prefixed with I2_. Please put a global $I2_WHATEVER; at the beginning of the function in which you need to make use of the variable.

Error Handling

If your function can continue in a reasonable fashion, use the d() method to output a debug message and continue. The debugging levels are as follows:

1. Major errors being concealed

    - Failing to load a template or file

2. Minor errors being concealed

    - Incorrectly formatted queries
    - Attempting to set an invalid style
    - Having to use the default value of something in an INI file

    If a debug would be level 1 or 2, carefully consider throwing an exception instead of trying to continue.

3. Strange events which are almost certainly abnormal

    - Queries returning errors

4. Strange events which are probably abnormal
   - Queries timing out
   - Weird combinations of session variables (user but no password, etc)

   Levels 3 and 4 will usually be errors thrown and dropped to debugging by the error handler.

5. Strange events which are not necessarily abnormal
   - No 8th-period activities in the given future

   Things 5 and above will probably always be logged to a file.

6. Less frequent, but normal, events
   - Construction of null objects
   - User not having CSL files but attempting to access them anyway
   - User's CSL auth failing with their LAN username and password

7. Routine module usage and high-volume stuff
   - LDAP queries
   - MySQL queries
   - Group-status messages ("user is an XYZ admin")

8. Server connections
   - LDAP and MySQL connections
   - Authentication babble

9. Module loading
   - "Loading Module XYZ"

If you cannot continue, throw an Exception. This will be caught by the Iodine built-in error handling methods and dealt with.

2.2.4 Smarty TPL

Smarty TPL is used to keep apart the way Iodine looks and the way it acts. Any and all HTML should be kept in TPL files in a directory corresponding to your module. These should be outputted using Display's **disp** method, called on the Display object your module is passed.

2.3 What to do if something goes wrong

2.3.1 Blame LDAP

It's probably LDAP's fault. Make sure that you can access LDAP on the Iodine server's command line with these commands:

- ldapwhoami -x
- kinit
- ldapwhoami

Next make sure that **phpinfo**() displays "SASL support".

2.3.2 Blame PHP

It's probably PHP's fault. Do you have a version of PHP with mcrypt and SASL support?

2.3.3 Blame MySQL

It's probably MySQL's fault. Is the mysql server up and running?

2.3.4 Blame the users

It's probably the user's fault. Do they have an IQ above 40?

2.3.5 Blame yourself

Only use this debugging technique if all the others fail. It is possible that the error is a result of your own coding mistake. Now go and fix it.

2.4 Installing Iodine on a new system

Iodine requires the following technologies:

- Apache 2
- PHP5, preferably via mod_php. PHP must be built with mcrypt, mysql, ldap, and sasl support enabled. It also needs to support persistent sessions, preferably imap, inifiles, and ctype support. You also almost certainly want ssl and threading

- mod_rewrite
- Smarty TPL
- Cyrus SASL
- OpenLDAP version 2.2 and up
- Heimdal Kerberos 5 version 0.7.2+
- Squirrelmail (or Zimbramail) for inline email
- LaTeXfor Eighth-Period formatting and display
- OpenAFS or another filesystem for the filecenter module

To install a new Iodine server:

1. Install Apache2 and get mod_php running. This is easy and very well documented.

2. Install a MySQL server. Create an iodine database, and a user which has full access to that database. Run each of the MySQL scripts in the mysql/ directory using the MySQL source command after selecting the iodine database. This will create initial empty tables. Use the base.sql file after all the others to create the seed data allowing Iodine to start running. Alternatively, use a dump of the real Iodine database.

3. Install a Heimdal Kerberos 5 server. The steps look something like this to initialize it:

   (a) Install Heimdal

   (b) Edit /etc/krb5.conf and change the default_realm to WHATEVER.COM, and set up the realms section and domain-realm mapping

   (c)
   ```
   kadmin -l
   kadmin¿ init WHATEVER.COM
   kadmin¿ add myuser
   kadmin¿ add myuser/ldap
   kadmin¿ add -r ldap/ldap.whatever.com
   kadmin¿ ext -k /etc/krb5.keytab ldap/ldap.whatever.com
   ```

   This places a keytab with the right service principal for the OpenLDAP server you are about to install in the default system keytab.

4. Ensure that reverse DNS is working properly
5. Install an OpenLDAP server with SASL support. This is the most tricky part of the whole installation. The steps are as follows:
   (a) Install OpenLDAP
   (b) Place the iodine.schema file from /ldap in your /etc/ldap/schema/ directory
   (c) Edit slapd.access to correspond to your new Kerberos realm
   (d) Edit slapd.conf and make the following changes:
      - Ensure the inetorgperson.schema file is included
      - Include the iodine.schema file after it
      - schemacheck on
      - sasl-host ldap.whatever.com
      - sasl-realm WHATEVER.COM
      - include slapd.acl
      - include slapd.access
      - Create the database as in the sample slapd.conf file.

   (e) Ensure that slapd starts with "-h ldap://ldap.whatever.com" or use SSL if you have the skill
   (f) Edit /etc/ldap/ldap.conf and set URI ldap://ldap.whatever.com
   (g) Populate the database with base.ldif or a real Iodine dump
   (h) Test by running the following commands:
      - kinit myuser
      - ldapwhoami -x
      - ldapwhoami -Y GSSAPI

      If the last command does not return uid=myuser,ou=people,dc=tjhsst,dc=edu something is wrong
6. Install a mail server/client as desired
7. Install filesystems as desired
8. Configure mod_php to respond to the .php5 suffix with the following line in httpd.conf:

```
AddType application/x-httpd-php .php5
```

9. Copy Iodine to a directory in which your Apache allows overwrites via mod_rewrite. Using mod_userdir is recommended. If you do, you may use "./setup home" to make your life much easier

10. Copy .htaccess.server to .htaccess and config.server.ini to config.ini

11. Edit config.ini so that all paths and most especially www_root are correct. Check the MySQL and LDAP server hostnames.

12. Profit!

2.5 Future possibilities

- Fix bugs and improve performance. For example, right now searching triggers one LDAP query per space-delimited part of the query. This could be combined into one large LDAP query string for a dramatic speed increase.

- Put student directory information into Active Directory and get it out of the Systems Lab's hands once and for all.

- Implement teacher evaluation forms and such using the polls interface.

- Allow more student interactivity in the Intranet—club sites, etc. Make putting together a club web page possible within the Intranet.

- Let the Computer Team grader and other club-specific apps use the Intranet framework.

- Let teachers post news to their classes.

- Finish what was abortively started: a collaborative due-date-tracking program where students may post when they think an assignment's due.

- A bibtex-based bibliography generator supporting MLA and APA styles. This is easier than you might think.

- Sign up for the next year's classes on the Intranet!

- The list goes on and on…