PHP Coding Standards

Code Source

Do not use any libraries or code from external sources without prior approval.

1) Similar code may already exist

2) Licensing may be an issue

3) Legibility and reusability can be issue

4) Performance concerns

Conventions h2.

Indenting:

Use an indent of 4 spaces, with no tabs. Please indent your code.

Case

- Use "lowerCamelCase" style (lowercase lettering on initial

words and capitalization on subsequent words) to name

functions, methods, and variables

Variables

Case should never be used to differentiate between variable names. Every variable name in the current scope should be absolutely unique. Variable names should describe the content that they (will) contain, using either complete words or understandable abbreviations.

Functions

No underscores except in the event handlers. Try to avoid abbreviations. Many programmers have a nasty habit of overly abbreviating everything. This should be discouraged. PHP functions are equivalent to a spoken language's verbs. Function

names, therefore, should be action oriented. They should also be defined in the present tense.

- Use "UpperCamelCase" style (capitalization on all words) to

name classes

- Define constants with uppercase letters and underscores

Constants:

SCREAMING_CAPS

Ex: define ("GLOBAL_CONSTANT","1");

Function Calls

Functions should be called with no spaces between the function name, the opening parenthesis, and the first parameter, spaces between commas and each parameter, and no space between the last parameter, the closing parenthesis, and the semicolon. Here's an example:

$var = foo($bar, $baz, $quux);

Function Declarations

Function declarations follow the "one true brace" convention:

function fooFunction($arg1, $arg2 = '')

{

if (condition) {

statement;

}

return $val;

}

PHP Code Tags

Always use <?php ?> to delimit PHP code, not the <? ?> shorthand.

HTML

HTML Code should not be found anywhere in our code, other than the smarty templates. Mark places where you find it with a TODO, as well as any plain text outputted to screen. We will convert these to a multiple language format in next version.

Control Structures

- Place a single space between the control keyword (if, for, while, switch, etc.) and opening parenthesis to distinguish control statements from function calls
- Always use curly braces—even when technically optional (i.e., avoid PHP's alternative syntax for control structures, except the ternary operator noted below)
- Include break; after all switch case statements
- Use the Allman/BSD style for indentation and layout (braces appear alone and surrounding the indented code)
- Always use elseif (one word) in lieu of else if (two words)

Example:

```
if ((condition1) || (condition2)) {
action1;
} elseif ((condition3) && (condition4)) {
action2;
} else {
defaultaction;
}
```

Ternary Operators

Can be used for simple true/false/action checks.

($result)? echo "<result>success</result>":echo "<result>failure</result>" ;

Database Tables and Fields

- Plural, lowercase words separated by underscores:
- • items
- • item_specifications
- Do NOT make calls directly to native database functions (ie:

mysql_insert_id, mysql_result,etc). Use ONLY adodb functions.

Smarty plugins

- function.smarty_plugin.php

Directories, File Structure

- Do not create new directories or class, library files without first discussing

with team. Most functions,classes belong in existing files.

- New plugins, and files should follow existing layout and naming.

=h2. Security =h2.

Passwords

- Never hardcode passwords into any script
- Never put live data passwords into SVN
- Always get your database passwords from included csp config.

Configuration

- Never hard code paths to any file – use configuration files.

Database

- Do not distribute database information. (via email, download or otherwise)
- Do not connect to production server from home (only dev).
- Do not run test or live queries on production data.

User Data

- All Credit Card information should be encrypted.
- User data should be obfuscated before being used in development

environments.

=h2. Code Documentation =h2.

Follow PHPDOC.org documentation style.

Subversion

Include comments for all commits to SVN. If you are fixing a bug, included the TICKET

ID in the Subversion Comments

TODO

Anytime you need to make comments regarding future modifications or requirement put

a //TODO: comments. These will show up in PHPEclipse and give us an idea of things

left unfinished (but functional). Example:

// TODO: This code should be re-examined for increased performance

CODE CLEANUP

Where you see standard rules broken -> MARK CODE WITH A TODO

PHP Comments

Use C-style (/* */) comments for multi-line statements and C++ comments (//) for single-

line statements

Database conventions

- Tables will always be named singular names for most tables, except when dealing with a clear collection. Thus the foreign key relationship will hold true to table_id form.

- Tables will be grouped by prefixes of primary related tables (itemProperties related to table item)

- Additional prefix grouping will be used for tables related to a particular functionality - "workspace", "sys" - this is only because there isn't a parent table, but the tables are in fact related, but by functionality.

- Primary keys of all tables will be lowercase id, except in the case of composite keys (those will depend on the combination)

- Foreign keys will be table_id (so if primary table is "site", and its primary key is "id", when "id" is referenced in other tables, they will reference "site_id".

With few exceptions, the standards we are following are documented here:
http://weblogs.asp.net/jamauss/articles/DatabaseNamingConventions.aspx Database Naming Conventions and
http://weblogs.asp.net/jamauss/archive/2004/05/10/129448.aspx DNC Deux