Online Registration System

Version 1.9

January 3, 2024

Contents

1. Overview

The online registration system provides a way for users to register and pay for an event via a web page.
It is licensed under the Open Software License version 3.0.
It uses PayPal to collect the money. In the PayPal web site, the user can use either a PayPal account or a credit card (no PayPal account needed).
Features:

- Optional Attendance registration for one event
- Optional sale of merchandise (T-Shirts, etc.)
- Optional payment of membership (Single or Family)
- Any combination of Attendance, Merchandise and/or Membership
- Registration for multiple people
- Option to not share personal information with others
- Options to indicate Gender, Child, and/or Student
- Optional custom Status values to support things like a Waiting List (handled manually, not automatically by the system)
- Optional Hospitality options: Need, Offer, Smoking, Cats, Dogs, Car Pooler, etc.
- As many custom data fields as needed, in various formats (number, decimal, text, radio buttons, selection lists)
- Optional support for multiple phone numbers and types (home, cell, etc.)
- As many Attendance options as needed
- As many Merchandise options as needed
- Optional discount for Early Registration or fee for Late Registration (date is handled automatically)
- Optional disable of Early/Late Registration Discount/Fee for any Attendance item(s)
- Optional discount for other purposes (organization membership, etc.)
- Optional donation field
- Optional payment of deposit instead of entire amount
- Optional payment of PayPal fee
- Optional use of custom PayPal Payment pages
- Optional return to your web site from PayPal (with creation of confirmation page)
- Optional shutdown of registration at a specified date (date is handled automatically, with creation of registration closed page)
- Automatic updating of registration status for PayPal Refunds
- Confirmation email is sent to user and to registrar, for Registrations and Refunds
- If hospitality is filled in, email is sent to hospitality coordinator
- Separate page (NOT password protected) to display just the names of the Paid registrants
- Separate page (password protected) to display all registrations and download in Excel CSV format
- Separate page (password protected) to edit or delete registrations
- Separate page (password protected) to create database table and the HTML registration page via one-button clicks

- Optional reference to this Online Registration System in registration page footer
- Logging of all registrations and paypal transactions
- Error checking for invalid or missing data
- Restricting web access to sensitive files
- Setting of PHP error options

2. Usage Overview

- User fills in the form, then clicks the Pay button
- The data is checked for missing or invalid data. Errors are displayed in a popup window, and problem fields are outlined in red.
- User corrects all error and re-submits the form.
- The data is sent to a PHP file in your web site. It is re-verified to try to prevent hacking. Error messages are displayed in a new web page.
- If verification passes, the data is saved in a database as a Pending registration, and the user is sent to the PayPal web site.
- User pays with either a PayPal account or a credit card.
- PayPal sends the payment information back to a PHP file in your web site.
- The database is updated to indicate registration is Paid.
- Emails are sent to the user and the registrar.
- If hospitality fields were used, email is sent to the hospitality coordinator.

If the user does not complete the payment (cancels out of PayPal), the registration remains in the Pending state.

The system provides an option to tell PayPal where to return after payment. The system can generate a confirmation page which can be customized.

If the user who has not completed payment attempts to register again, a NEW (Pending) registration is created. The user can re-register an unlimited number of times until payment is completed. When a registration finally reaches the Paid state, no more registrations for that user are permitted (based on name and email address).

A page is provided that lists the names of all Paid registrations. Use of this page is optional.

The registrar can login to special page to look at the registrations, and optionally download them in Excel CSV format.

The registrar can login to special page to edit registrations to correct errors.

If a registered user cannot attend the event and needs a refund, if they paid using PayPal, the PayPal web site can be used to issue a refund, within 60 days of payment. After 60 days, a refund must be issued via check instead of PayPal.

On and after a specified date, the online registration page can forward to another page to indicate registration is closed. The system can generate a registration closed page which can be customized.

After the event, the PayPal web site can be used to transfer money from PayPal to the linked bank account.

3. Requirements

- PayPal Seller account
- Web server with PHP scripting (version 5.x, 7.x, or 8.x) and MySQL databases
- Someone with enough HTML and web server knowledge to create and modify files in your web site, including creating a MySQL database.

4. Setting up a New Registration System

Setting up a new registration system requires several steps:

- Create a PayPal Seller (if you don't already have one)
- Copy the registration system files to your web site
- Create a MySQL database, user, and password in your web server
- Customize the olr-config.php file for your event. Make sure the PayPal fee data matches the current fees for your account.
- Go to the builder page in the system and click the buttons to create the database table and to create a preliminary version of the registration HTML page
- Make a copy the registration HTML page and customize the layout and text as desired. Modify the CSS file as desired.
- If the return URL from PayPal feature is used, customize the online-reg-confirm.html page as desired.
- If the automatic shutdown feature is used, customize the online-reg-closed.html page as desired.
- For WordPress or other Content Management Systems (CMS), incorporate the contents of the registration HTML page and the CSS file into a new Page created in your CMS system.
- Test the system

Each step is detailed in the following sections.

4.1. PayPal Seller Account

The system requires a PayPal Seller Account. If your organization does not already have such an account, go to the PayPal web site (http://www.paypal.com) and create one.
One PayPal account can be used for all of your events.

If you are a non-profit organization, you have to contact PayPal to request non-profit status. You will need to supply some paperwork to prove your non-profit status. Our dance organization is non-profit via its association with the Country Dance and Song Society (CDSS). Upon request, CDSS will send you a letter verifying that you are an affiliate of theirs, and therefore you inherit their non-profit status.

PayPal does have a fee for transactions, consisting of a small fixed fee, plus a percentage of the transaction. There is a discount for the percentage part of the fee for non-profit organizations. The fee does change over time.

To determine the fees for your account, login to your PayPal account. In the top menubar, click "Pay & Get Paid", then select "Money" in the drop down menu. On the left, click "Merchant fees". The fees for your account will be listed in the leftmost "PayPal" section.

Example: If the transaction amount is $100, and the fee is $0.49 + 3.49% (totaling $3.98), the buyer pays $100, but the seller only gets $100 - $3.98 = $96.02.

The registration page calculates the expected PayPal fee and displays it on the page, with an option for the user to add it to their total (e.g. $103.98 instead of $100), so your organization gets the full registration amount.

There are two variables in the configuration file to set the fixed and percentage values of the PayPal fee ($c_paypal_fee_fixed and $c_paypal_fee_percent , see later section about the configuration file details). The system uses these values to calculate the expected PayPal fee. Since PayPal does change their fees every so often, the configuration file variables should be checked against your current fees during registration setup. If these values are incorrect, nothing bad happens. You will just not receive the expected amount after PayPal deducts its fees. The payment confirmation that PayPal sends the system contains the actual fee and net amount, and the system updates the database with these values. Thus the system always has the correct financial information, even if the configuration file variables are incorrect.

PayPal provides several ways to customize the payment page, using options in the account Profile. The standard payment page has a text header/title which is set by the option "Business information" in the "My business info" section of the Profile. If you are using the standard payment page, you probably want to set "Business information" to your organization's name, since it will be used for all of your events.
In the "My selling tools" section of the Profile, the option "Custom payment pages" can be used to define custom payment pages with your own graphics. You could create a custom payment page for your organization, or you could create a custom page for each of your events. Each custom page has a user-defined Page Style name. The registration system provides a way to specify the page style to be used for the payment page.

4.2. Install Files

For a new installation:
Create a new folder, named appropriately for your event, in your web site and copy all the files in the "website" folder of this system to the new folder. If the web site uses WordPress, the new folder can be under the top level WordPress folder, or under the content folder "wp_content".

For an existing installation:
See the README.txt file for how to update an existing installation, and if that is even recommended.

4.3. Setup MySQL Database

The registration data is saved in a MySQL database in the web server. You will have to create the database and the user/password for the database. All the data for one event is saved in one database table. A database can contain many tables, so you only need one database for all your events.

If you have used the registration system in the past so you have previously created a database and user (and they still exist), you can use the same database and user for your new event. Get the database name, user name, and user password from the previous event, and skip the rest of this section.

If you have not used the system before, or you want to use a separate database and/or user for the new event, proceed with the instructions in this section.

4.3.1. Create the Database

Most web hosting services provide a control panel to manage your web site. These instructions are based on the popular "cPanel" control panel. If your web server uses a different control panel, hopefully you can figure out how to apply these instructions to your control panel.

1. Login to your web host control panel.
2. Scroll down to the Database section and click the "MySQL Databases" icon to open it.
3. There is a section near the top to create a new database. Enter a name for the database (e.g. onlinereg, etc.) and click the "Create Database" button. The database name might be case sensitive, depending on the web host. Note that the system will probably add a prefix to your database name, consisting of your web host login name followed by an underscore.
4. A new page should be displayed with a success message. Click the "Go Back" link on the page to go back to the first page.
5. Scroll down to the Users section. Enter a name and password for a new user and click the "Create User" button. Note that the user name might be limited to 7 or fewer characters. Note that the system will probably add a prefix to your database name, consisting of your web host login name followed by an underscore. If you want to use an existing user instead of creating a new one, skip down to step 7.
6. A new page should be displayed with a success message. Click the "Go Back" link on the page to go back to the first page.
7. Near the bottom of the page is a way to associate users with databases. Select the new database and the new user, then click the "Add" button.
8. A new page should be displayed, asking for the permissions for the new user. Check the "All Privileges" box, then click the "Make Changes" button.
9. A new page should be displayed with a success message.
10. Make a note of the host name (typically "localhost"), database name, user name, and user password for later use when customizing the olr-config.php file. Note that the system might prefix the database name and the user name with the login name of your web hosting account. That is part of the name and must be included in the values in the olr-config.php file.

4.4. Customize Security and Error Options

For security reasons, it is recommended to NOT display to the user certain types of files and to NOT display detailed error messages. Several files are included in this application to make it easy to control file access and PHP error reporting options:

- .htaccess-cgi – restricts access to certain files, for Apache web servers
- .htaccess-apache – restricts access to certain files, and sets PHP error options, for Apache web servers
- .user.ini – sets PHP error options
- web.config – restricts access to certain files, for Windows IIS web servers

There are two common operating systems used for web servers: Unix/Linux and Windows. There are several common applications/programs used for web servers: Apache, Lightspeed, NGINX, and Microsoft IIS. Apache can be used with both Unix/Linux and Windows operating systems. Microsoft IIS can only be used with the Windows operating system. In addition, there are different ways that PHP can be run on a web server. The method used to restrict file access and to set PHP error options depends on the web server application and the way PHP is run on the web server.

Browse to the olr-zzzphpinfo.php file. This will display detailed information about your web server PHP installation.
The PHP Version is displayed at the top of the page. The online registration system has been tested with versions 5.6.x, 7.4.x, and 8.1.x. If your web server is NOT running one of these versions, you might be able to change the PHP version used by your web site (see next section).

In the PHP Info page, in the first table of data, look for the variable "Server API". Make a note of the value of the variable.

In the PHP Info page, scroll down to the "Core" table. Look for the variable "user_ini.filename".
If it is not ".user.ini", rename the ".user.ini" file in the online registration folder to the value of this variable. In the instructions that follow, this file will be referred to as ".user.ini". If you have renamed it, use the new name in place of ".user.ini".

In the PHP Info page, scroll down to the "Environment" table. Look for the variable "SERVER_SOFTWARE". Make a note of the value of the variable.

If the "SERVER_SOFTWARE" is "Apache" or "Lightspeed" and the "Server API Settings" is "CGI" or "CGI/FastCGI" the .htaccess and .user.ini files will be used by PHP.
Make a copy of the file ".htaccess-cgi" and rename the copy to ".htaccess".
The new ".htaccess" file should not need to be changed.
Customize the PHP error options (see below) in the file ".user.ini".

If the "SERVER_SOFTWARE" is "Apache" or "Lightspeed" and the "Server API Settings" is "Apache" only the ".htaccess" file will be used by the system (NOT ".user.ini").
Make a copy of the file ".htaccess-apache" and rename the copy to ".htaccess".

Customize the PHP error options (see below) in the new ".htaccess" file.

If the "SERVER_SOFTWARE" is "IIS", the "web.config" and ".user.ini" files will be used by the system.
The "web.config" file should not need to be changed.
Customize the PHP error options (see below) in the file ".user.ini".

If the "SERVER_SOFTWARE" is "NGINX", the .htaccess file is ignored,
and there is no way to customize the configuration.

The PHP error options are contained in the ".user.ini" file and also in the ".htaccess-apache" file. The default values are set for a production environment: errors are written to a log file, and NOT displayed to the user. To see the errors, you have to look at the error log file. This is inconvenient during development, so you might want to change the settings to display errors to the user during development, then turn off display of errors when ready to go to production. There are several error options, but there are only three that you should consider changing.

Here are the error options that you might want to change in the ".user.ini" file:

- display_startup_errors = off
- display_errors = off
- error_log = php-errors.log

When the "Server API Settings" is "Apache" or "Lightspeed", the error options are in the ".htaccess" file instead of in the ".user.ini" file. Here are the options in that file:

- php_flag display_startup_errors off
- php_flag display_errors off
- php_value error_log php-errors.log

For both files, the display_xxx value is **on** to display errors (for development) and **off** to hide errors (for production).
The "error_log" value is the name of the error log file. You can change this if you want, but it should end with ".log" to be protected from viewing by the users. The value must match the $c_error_logfile parameter in the olr-config.php file, so if you change it, be sure to change it in both files.

4.5. Choose PHP version (OPTIONAL)

Starting with version 1.7, the system works with PHP versions 5, 7, and 8 (there apparently is no version 6). Therefore**, the system should work on most web servers as is, with no customization needed.**

The olr-zzzphpinfo.php file (see previous section) will tell you what version your server is running by default. If you want to use a different version of PHP, it is possible to change it, at two different levels.

Your web host control panel probably has an app to select the PHP version. That will apply to the entire web site. Login to your web server control panel and look for an item named "PHP Configuration"

(might be at the bottom in the "Advanced" section). That will allow you to change the PHP version used by your web site. Choose the latest supported version (up to 8.1.x).

With the Apache or Lightspeed or other web server that uses the .htaccess file (see previous section), it is possible to set the PHP version for each folder. The supplied .htaccess file has examples of this, but they are commented out. The lines to set the PHP version are similar to:

```
<IfModule mime_module>
  AddHandler application/x-httpd-alt-php81 .php
</IfModule>
```

The "alt-php81" is the PHP version number in the web server, and varies with the setup your web server. The control panel app which allows you to change the PHP version might show the version number. When you do change the PHP version, an entry like the above is probably placed in the .htaccess file in the root folder of your web site. You can look at that to get the version number, or copy those lines to your local .htaccess file.

4.6. Customize PHP Configuration File olr-config.php

The olr-config.php file is the configuration file for the system. It contains information like the MySQL database name, user, and password, email addresses for the registrar and hospitality coordinator, and information about the data fields you are using. It must be customized for your event. The file is a plain text file using the PHP language format. You can edit it with any text editor. The default file provided has ALL the possible fields defined, so your editing will involve a lot of deleting unwanted data, and modifying existing data. Examples of other configuration files are in the examples folder.

NOTE: If there are options you will not use, DO NOT DELETE THEM FROM THE FILE. Set the option to a value that indicates it is not used, typically null or "" (see details below).

4.6.1. Data Fields

The registration system supports several different types of data fields. There are some pre-defined data fields used for specific purposes. In addition any number of custom fields can be defined.

The data fields are defined in the olr-config.php file. Each data field is defined by several attributes:

- label = the text displayed to describe the data
- abbrev = abbreviation of the description, used in the registration list page and its Excel download. It is best to use underscores as a separator instead of spaces or dashes.
- name = unique name for the data field, used as the "name" attribute of the HTML elements in the registration page, and also as the table column name in the database. The system pre-pends the name with a prefix based on the type of the data. The MySQL database does NOT allow dashes or spaces in column names; use underscores as a separator. It is best to use all **lowercase**.
- type = the type of the data. Supported values are (always use **lowercase**):
  "state" for 2-letter State abbreviation (prefix = "txt_")
  "zip" for zipcode (prefix = "txt_")
  "email" for email address (prefix = "txt_")
  "phone" for phone number (prefix = "txt_")

"check " for yes/no values (checkboxes) (prefix = "chk_")
"num" for whole numbers (prefix = "num_")
"dec" for decimal numbers, like money (prefix = "dec_")
"text" for any other text value (prefix = "txt_")

- size = For text data types, this is the maximum number of characters. For numbers, this is the number of digits. Use 2 for small numbers like number of people. Use 6 for dollar values. For checkboxes, use 1.
- required = optional attribute to indicate the data field is required. Use required = 1 for required fields. For fields that are not required, either leave off the "required" attribute, or use required = 0. Note that "required" must be **lowercase**.

Make a list of all the data fields you need in the registration form, and their attributes, and use that to customize the olr-config.php file.

The system supports registration for multiple people at one time, but with just one address, phone(s), and email for everyone. You can create separate data fields for each person (street1, street2, etc.), but the system will not associate the data fields with the people. You will have to do that on your own.

Our organization uses the registration system for two different types of events, so it is designed to accommodate both types of events. Both types of data are included in the olr-config.php file. You can remove any/all of these fields you don't need.

- An event in the city, using a hired band and caller. For this type of event, local people offer hospitality in their homes for out of town attendees. The registration page can include fields for both requesting and offering hospitality.
- An event in a remote location where housing is provided in cabins. The registration page can include fields for cabin choices. No external band or caller is used; the attendees provide all the talent for the event. The registration page can include special fields for musicians and callers to indicate their talents, so a band and caller schedule can be created for the event.

4.6.1.1. Pre-Defined Fields

Some fields have special meanings, so must use a pre-defined "name" for the system to work properly. The "email1" field is required. The other fields are optional, but if they are used, they must use the pre-defined name. All these field names must be **lowercase**.

- email1 = email address of first or primary person
- emailxx = email address of additional persons must start with "email"
- phone = the name for all phone fields should start with "phone" (e.g. phonecell, etc.)
- street = the name for all Street fields should start with "street"
- city = the name for all City fields should start with "city"
- state = the name for a State field must contain "state"
- zip = the name of a Zipcode field must contain "zip"
- housing = the name of all fields dealing with housing (cabins, hotel rooms, etc.) must start with "housing"

- hosp_req_beds, hosp_req_num = the number of beds being requested, or the number of people requesting home hospitality. You can use either or both fields. The number of beds eliminates the host trying to guess who is willing to share a bed.
- hosp_req_xxx = the name of all other fields dealing with hospitality requests must start with "hosp_req". You can define as many as you want for things like gender, cats, dogs, smoking, allergies, etc. (e.g. hosp_req_men, hosp_req_women, hosp_req_dogs, hosp_req_smoker, hosp_req_allergies, etc.)
- hosp_offer_beds, hosp_offer_num = the number of beds being offered, or the number of people that can be housed for home hospitality. You can use either or both fields. The number of beds is probably better to use, since the host does not know which guests are willing to share a bed.
- hosp_offer_xxx = the name of all other fields dealing with hospitality offers must start with "hosp_offer". You can define as many as you want for things like gender preference, cats, dogs, smoking, etc. (e.g. hosp_offer_prefer_men , hosp_offer_prefer_women, hosp_offer_cats, hosp_offer_dogs, hosp_offer_smoker, etc.).
- men_dancers = number of men attending the event (used for gender balancing)
- women_dancers = number of women attending the (used for gender balancing)
- mus_name = name of musician
- call_name = name of caller

4.6.1.2. Phone Number Fields

Collecting phone number data can be tricky, especially when at least one phone number is desired to be a required field. Some people have multiple phone numbers (home, cell, etc.). Some people just have a home phone. Some people have just a cell phone. Trying to deal with these variations and have at least one required phone number field is difficult. The following discussion references the configuration parameters described in the following Config File Details section of this document.

The simplest scenario is to collect one phone number, without specifying the number type (home, cell, etc.). Let the user decide which of their numbers to use. For this scenario, a generic "Phone" field can be used in the $c_aryFieldDefs, and it can be set to be required.

You can include multiple phone fields in $c_aryFieldDefs, but making one of them required is tricky. If you define the fields as specific types, like "Home Phone" and "Cell Phone", setting one of them to be required will be a problem for users having just one phone number of the type that is NOT required (e.g., requiring "Home Phone", but user just has a cell phone). One way around this is to add some text to the page to tell the user to use the same number for both fields if they only have one number. Another workaround is to make the phone names more generic, like "Phone 1" and "Phone 2", or "Primary Phone" and "Secondary Phone", and setting just the first one to be required. Yet another option is to make a generic "Phone" field required, and provide a text "Phone Type" field for them to fill in "Home", "Cell", etc.

If it is desired to collect one or more phone numbers AND the type of the number AND require at least one number, the configuration item $c_aryPhone_Fields can be used instead of defining the phone

field(s) in $c_aryFieldDefs. The $c_aryPhone_Fields data consists of an array of phone number data. For each number, you can specify a set of Radio Buttons to specify the type of the number (Home, Cell, Work, etc.). You can define as many numbers as you want to collect, with as many options as you want for each one. For each number, the registration page will have a text field for the number itself, and a set of Radio Buttons for the user to specify the type of the number. The database stores both the number and the type of the number. For each number that you want to be Required, include the attribute "required" => 1.

If defining the phone field(s) in $c_aryFieldDefs is sufficient for your needs, the value of $c_aryPhone_Fields should be set to "null" (without the quotes).

4.6.2. Config File Details

Here is a detailed description of the parameters in the olr-config.php configuration file.

date_default_timezone_set (xxx)
change to your local timezone (see comments in the file)

$c_olr_version
the version of the online registration software. Do not change.

$c_test_mode
set to 0 for normal operation, or 1 to use PayPal Developer Sandbox for testing the system without spending any real money (see later section on Testing).

$c_email_pending
set to 1 to send an email to the webmaster for each "Pending" registration (before PayPal), or to 0 to not send that email

$c_email_pending_user
set to 1 to send an email to the USER for each "Pending" registration (before PayPal), or to 0 to not send that email. You probably do NOT want to send these emails.

$c_EventName
name of your event, used in the registration HTML page and email messages

$c_EventAbbrev
abbreviation for the name of your event, used in the Transaction ID sent to PayPal

$c_online_reg_fname
name of the registration HTML file that will be generated by the system. This should probably NOT be the final name you want to use in the web site. You will probably want to customize the generated file, so you should use a different file name for the generated and customized/final versions, so the page generator does not overwrite your customizations.

$c_reg_system_info_in_footer
set to 1 to include a line with the online registration system name and version at the bottom of the registration page. This is not required by the license, but is appreciated by the developer.

$c_notify_url
full URL of the system's pp-ipn.php file in your web site

$c_return_url
full URL of the web page you want the user to return to after payment. If this is blank (""), PayPal does not give the user the opportunity to return to your web site. If this is not blank (""), PayPal will generate a button to return to your web site (this URL) in its payment confirmation page. PayPal does NOT automatically forward to this URL. The user has to click the button to return to your web site. The button is near the bottom of the PayPal confirmation page, so the user usually has to scroll down the page to see the button. The URL should be some sort of confirmation page for the registration. If the value is not blank, the system will create a default confirmation page named online-reg-confirm.html when the online registration is built. If there is already a file named online-reg-confirm.html, the system will NOT overwrite it. Therefore, you can let the system create the default file, and you can customize it as desired without worrying about losing the customizations.

$c_paypal_page_style
the name of the custom PayPal payment page style to use for payments. Leave the default empty string to use the standard PayPal payment page.

$c_excel_filename
default name of file for the Excel CSV download of the registrations (the filename can usually be changed during the download)

$c_payment_email
email address for your PayPal seller account

$c_webmaster_email
email address for errors and optional pending registration emails

$ g_print_reg_login_pwd
password for the olr-print-regs.php page that displays the registration data (case sensitive)

$ g_edit_reg_login_pwd
password for the olr-edit-regs.php page that edits the registration data (case sensitive)

$ g_build_reg_login_pwd
password for the olr-build-all.php page that builds the database and registration HTML page (case sensitive)

$c_dbhost
MySQL database host name (typically, "localhost")

$c_dbname
MySQL database name (might be case sensitive, depending on web server)

$c_dbusername
MySQL database user name (might be limited to 7 characters, might need web site user account name prefix)

$c_dbpassword
MySQL database user password (might need web site user account name prefix)

$c_dbtable
MySQL database table name to use for the data (underscores not allowed)

$c_db_table_def_file
name of the file used to save the MySQL table definition. This file is created by the system, for reference only. It is not actually used by the system. The database table is created directly by the builder page, instead of using this file. To prevent web site users from seeing the contents of the file, it is now implemented as a php file, with the contents in a comment. If the filename you specify does not end with .php, that is automatically added to the filename when it is created. Any attempt by a web user to view this file will simply result in a blank page. If you want to see the file, you have to view it via your web site control panel, or use FTP to copy it from the web server to your computer.

$c_custom_statuses
any custom Status values you want to use for the registrations. This can be a single value (e.g. "Waitlist"), or a comma-separated list of values (e.g. "Waitlist,Cancelled"). See the section on Custom Status Values below for more details.

$c_max_persons
maximum number of persons that can register at one time. A set of name, gender, child, and student fields will be generated for each person.

$c_UseChildData
If set to 1, a "Child" checkbox will be generated for each person except the first person. If set to 0, the "Child" checkbox is NOT generated.

$c_UseChildAge
If set to 1, an "Age" text box will be generated for each person except the first person. If set to 0, the "Age" text box is NOT generated.

$c_UseStudentData
If set to 1, a "Student" checkbox will be generated for each person. If set to 0, the "Student" checkbox is NOT generated.

$c_UseGenderData
If set to 1, a set of "Gender" radio buttons will be generated for each person. If set to 0, the "Gender" radio buttons are NOT generated.

$c_Use_Do_Not_Share
If set to 1, a checkbox will be included in the registration page to indicate the personal information should not be shared with the other registrants. If set to 0, the checkbox is not included.

$c_UseDonation
If set to 1, an optional field will be included in the registration page for people to make donations. If set to 0, the donation field is not included.

$c_DepositPrice
Dollar amount of the deposit required, if/when allowing people to pay a deposit instead of the whole amount. Set to 0 to disable the deposit field.

$c_DepositDueDate
Due date of the balance of the payment after the deposit is paid. This is optional, and can be in any date format. It is simply displayed in the registration form. It is not used for any calculation or action. Set to empty string "" if not displaying this.

$c_UsePayPalFee
If set to 1, a checkbox will be included in the registration page for the user to indicate they will pay the PayPal fee. If set to 0, the checkbox is not included.

$c_checkbox_before_label
This controls the placement of the label for checkbox data fields. Setting it to 1 puts the checkbox before (to the left of) the label. Setting it to 0 puts the checkbox after (to the right of) the label.

$c_radio_before_label
This controls the placement of the label for radio button data fields. Setting it to 1 puts the radio button before (to the left of) the label. Setting it to 0 puts the radio button after (to the right of) the label.

$c_aryFieldDefs
Definitions of the data fields you are using. The definition of each field has the following format (use a comma after each field):

array("label" => "xx1", "abbrev" => "xx2", "name" => "xx3", "type" => "xx4", "size" => xx5),
where the attributes (label, etc.) are as defined above in the section about Data Fields.

$c_text_line_width
Maximum length of single-line text boxes. Any text field shorter than this is implemented as a HTML "<input type='text' …> element, which is displayed as a single line. Any field with a length longer than this will be implemented as a HTML "<textarea>" element, whose width is this maximum line width, and whose height is enough to contain the full number of characters.

$c_name_max_length
Maximum length for the First Name and Last Name fields.

$c_aryPhone_Fields
Definition of telephone number fields to be used in the registration page when it is desired to know both the number and the type (see the previous section about Phone Number Fields). The definition of each phone number has the following format (with example data values; use whatever you want):

```
array("label" => "Phone 1", "abbrev" => "Phone1", "name" => "phone1", "size" => 12, "required" => 1,
"items" => array (
  array("label" => "Home", "value" => "Home", "default" => 1),
  array("label" => "Cell", "value" => "Cell)
 ),
("label" => "Phone 2", "abbrev" => "Phone2", "name" => "phone2", "size" => 12, "items" => array (
  array("label" => "Home", "value" => "Home"),
  array("label" => "Cell", "value" => "Cell", "default" => 1)
 )
)
```

where the attributes (label, etc.) are as defined above in the section about Data Fields.
Each top-level array item is a set of data for one phone number. The example size of 12 is the minimum for the phone number format 123-456-7890. The "items" attribute defines the data for radio buttons to specify the type of phone number (e.g. Home, Cell, Work, etc.). The "label" is the text displayed next to the button. The "value" is the value stored in the database for the phone number type. Use the optional "required" attribute to specify which (if any) fields are to be required. If the "default" attribute is included for a type/button, that item will be selected by default. If no "default" attribute is supplied, the first type/button will be selected by default. The "size" attribute applies to both the phone number field and the phone number type fields. Be sure the "size" is large enough to accommodate both values, or make sure the "item" values (e.g. "Home", "Cell", etc.) are shorter than the "size". If the phone data is specified in the $c_aryFieldDefs section, set this option's value to "null" (without quotes) instead of an array.

$c_aryRadio_Fields
Definition of Radio Button Fields to be used in the registration page. Each Radio Button Field is a set of round buttons, where only one value can be selected. The definition of each button field set has the following format (use a comma after each field):

```
array("label" => "Choose a Color", "abbrev" => "color", "name" => "color", "type" => "text", "size" => 5,
"required" => 1, "items" => array (
  array("label" => "Fire Engine Red", "value" => "red"),
  array("label" => "Sky Blue", "value" => "blue", "default" => 1),
) )
```

where the attributes (label, etc.) are as defined above in the section about Data Fields.
Each top-level array is a set of related buttons in which only one can be selected at a time. The top-level attributes apply to the entire set of buttons. The "items" attribute defines the buttons in the set. The "label" is the text displayed next to the button. The "value" is the value stored in the database for the button. If the "default" attribute is included for a button, that button will be selected by default. If no "default" attribute is supplied, the first button will be selected by default, if the item is not "required'. If the item is "required" and there is no "default", no buttons will be selected, forcing the user to select one. If no Radio Buttons are needed, use the value "null" (without quotes) instead of an array.

$c_arySelect_Fields
Definition of Selection List Fields to be used in the registration page. Each Selection List Field is a drop-down list, where only one value can be selected. The definition of each list has the following format (use a comma after each field):

```
array("label" => "Choose a Color", "abbrev" => "color", "name" => "color", "type" => "text", "size" => 5,
"required" => 1, "items" => array (
  array("label" => "Fire Engine Red", "value" => "red"),
  array("label" => "Sky Blue", "value" => "blue", "default" => 1),
) )
```

where the attributes (label, etc.) are as defined above in the section about Data Fields.
Each top-level array is a selection list or "drop down" list of text values in which only one item in the list can be selected. The top-level attributes apply to the entire list. The "items" attribute defines the text items in the list. The "label" is the text displayed in the list. The "value" is the value stored in the database for the item. If the "default" attribute is included for an item, that item will be selected by default. If no "default" attribute is supplied, the first item will be selected by default. Because of how selection lists work, one item is ALWAYS selected. The user could easily overlook making a selection, causing the default selection to be used. If you want to force the user to make a selection, set the list as "required", do not specify a "default" item, and define the first item with both label and value as empty string (""). If no Selection Lists are needed, use the value "null" (without quotes) instead of an array.

$c_aryAttendance_Fields
Definitions of the attendance options for the event, but NOT including Early/Late registration values (see below for those). The definition of each option has the following format (use a comma after each field):

array("label" => "Full Event", "abbrev" => "full", "price" => 95, "no_early_late_reg" => 1),

where "label" and "abbrev" are as defined in the section above about Data Fields.
The "price" is the price of the option. You can define discount options (e.g. for members of your organization, etc.) by using a negative price. You can use a price of 0, to track the quantity of a special option (e.g. Children under 12 are free, but you want to know how many).
The “no_early_late_reg” field is OPTIONAL, and should only be used for attendance options that you do NOT want an early/late registration discount/fee applied to. For example, if you have an option for students or children which is already discounted, you might not want to apply the additional discount for early registration to those options. Another example of when to use the “no_early_late_reg” field is if you are providing registration for the full event, and also for individual sessions. You can use the “no_early_late_reg” field for each session so that an early registration discount is only applied to the registration for the full session, but NOT for the individual sessions. Note that an early/late registration discount/fee is only applied to attendance options with a price greater than zero, so there is no need to use the “no_early_late_reg” field for attendance options with a zero or negative price.

See the next paragraph about selecting one or more options per person.

$c_attendance_multi_select
specifies whether or not multiple attendance options can be selected per person.
When $c_attendance_multi_select = 0, the system limits the number of selected attendance options to one attendance option per person, except for discount options (zero or negative values). One or more discount options can be selected with the one non-discount option. If the user attempts to select multiple non-discount options per person, an error message is displayed.
When $c_attendance_multi_select = 1, multiple attendance options per user are allowed.
The header of the Attendance options section of the registration page indicates whether one or multiple selections per person are allowed.

$c_early_reg_discount
amount of the discount for early registration (negative value), if the registration is entered on or before the $c_early_late_reg_cutoff_date (see below). Use 0 if not using an early registration discount. The system automatically applies this discount once per person, no matter how many registration options they choose. See the section $c_aryAttendance_Fields above for how to disable the discount for one or more specific attendance options.

$c_late_reg_fee
extra fee for late registration, if the registration is entered on or after the $c_early_late_reg_cutoff_date (see below). Use 0 if not using a late registration fee. The system automatically applies this fee once per person, no matter how many registration options they choose.

See the section $c_aryAttendance_Fields above for how to disable the late fee for one or more specific attendance options.

$c_early_late_reg_cutoff_date
The last date for the early registration discount, or the first date for the late registration fee. The date format is "May 15, 2014". Use the empty string "" if not using early or late registration. For example, setting the date to May 15 means the early registration discount will be applied through the end of May 15, or the late registration fee will be applied on or after May 15.

$c_reg_shutdown_date
The first date that registration is no longer allowed. The date format is "May 22, 2014". On this and later dates, the online registration page will automatically forward to the online-reg-closed.html page. If you do not want to use this automatic shutdown feature, use the empty string "".

$c_aryMerchandise_Fields
Definitions of the merchandise options, like T-Shirts, etc. The definition of each option has the following format (use a comma after each field, except for the last):

array("label" => "T-Shirt", "abbrev" => "tshirt", "price" => 15)

where "label" and "abbrev" are as defined in the section above about Data Fields. The "price" is the price of the merchandise. Merchandise options are always implemented as text boxes, any quantity can be entered. The merchandise quantities are completely separate from the attendance quantites.

$c_member_single_price
Setting this to non-zero will include an option to purchase a single membership to your organization, at the specified price.

$c_member_family_price
Setting this to non-zero will include an option to purchase a family membership to your organization, at the specified price.

The following two fields are used for special events where you want information about the musicians coming to the event.

$c_max_musicians
maximum number of musicians to collect data for. This many sets of musician data fields will be generated in the registration HTML page. Use 0 if not using musician data.

$c_aryMusician_Fields
Definitions of the data fields for each musician. Use the value "null" (without the quotes) if not using this data.
The definition of each field has the following format (use a comma after each field):

array("label" => "Musician Name", "abbrev" => "Mus Name", "name" => "mus_name", "type" => "text", size => 20),

where the attributes are as defined in the section above about Data Fields.
The first field MUST be "mus_name", as shown above. Any number of other data fields can be included.

The following two fields are used for special events where you want information about the callers coming to the event.

$c_max_callers
maximum number of callers to collect data for. This many sets of caller data fields will be generated in the registration HTML page. Use 0 if not using caller data.

$c_aryCaller_Fields
Definitions of the data fields for each caller. Use the value "null" (without the quotes) if not using this data.
The definition of each field has the following format (use a comma after each field):

array("label" => "Caller Name", "abbrev" => "Call Name", "name" => "call_name", "type" => "text", size => 20),

where the attributes are as defined in the section above about Data Fields.
The first field MUST be "call_name", as shown above. Any number of other data fields can be included.

$c_paypal_fee_fixed
fixed part of PayPal fee (e.g. 0.49 for 49 cents). PayPal changes this over time, so use the current value for your account, as described in the earlier section about setting up a PayPal Seller Account.

$c_paypal_fee_percent
percentage part of PayPal fee (e.g. 0.0199 for 1.99% for non-profit, or 0.0349 for 3.49% for others). The file has a line for both regular and non-profit rates. Use "//" at the beginning of the line to comment out the one you do NOT want to use, and make sure "//" is not at the beginning of the line of the one you do want to use. PayPal changes this over time, so use the current value for your account, as described in the earlier section about setting up a PayPal Seller Account.

$c_pp_logfile
name of the log file for PayPal transactions. It should end with the ".log" extension for maximum protection from being viewed by anyone via the web browser (protected by the .htaccess file).

$c_reg_logfile
name of the log file for registrations (Excel csv format). It should end with the ".csv" extension for maximum protection from being viewed by anyone via the web browser (protected by the .htaccess file).

$c_error_logfile
name of the PHP error log file (must match the "error_log" parameter in the ".user.ini" or ".htaccess" file, see the section **Customize Security and Error Options** above).

$c_dump_form_debug
set to 0 for normal operation, or 1 for special debugging mode which prints all the data received when the registration page is submitted

$c_webmaster_email_errors
set to 1 for normal operation. If email is not working for some strange reason, setting this to 0 will display error messages that are normally emailed to the webmaster in the web page instead of emailing them.

The following are initialized in the Online_Reg_Config_Class constructor function, near the end of the olr-config.php file.There are two values, one for Production (when $c_test_mode = 0;) and once for the PayPal Developer Sandbox (when $c_test_mode = 1;)
Fill in the appropriate values.
$c_business_email
email of your PayPal seller account (Production and Sandbox)
$c_paypal_url
PayPal URL (Production and Sandbox) [DO NOT CHANGE]
$c_paypal_ipn_url
PayPal IPN URL (Production and Sandbox) [DO NOT CHANGE]
$c_registrar_email
Email of Registrar (use webmaster or developer for Sandbox)
$c_hospitality_email
Email of Hospitality coordinator (use webmaster or developer for Sandbox test mode). If you are not using Hospitality, this can be left blank ("").

4.6.3. Custom Status Values

The system has three built-in Status values for the registrations: Pending, Paid, and Refunded. These are handled automatically by the system. The $c_custom_statuses item in the config file allows you to add custom status values (e.g. "Waitlist"). The custom status values are NOT automatically handled by the system. The only time the system uses them is to add them to the Status list in the Registration Editor (see separate section about Editing Registrations), so registrations can be set to those status values manually.

One possible use for a custom status is for helping to maintain a Waiting List. There are many reasons why someone might be put on a waiting list: limited number of attendees, gender balance, etc. It is way too complicated for the system to try to manage a waiting list automatically. If the registrar determines that a Paid registration should be on a waiting list, the Registration Editor can be used to change the Status of that registration from Paid to Waitlist. If the registrar later determines the registration is allowed, the Registration Editor can be used to change the Status from Waitlist to Paid. If the registrar determines a Waitlist registration is not allowed, a refund can be issued. The system will automatically change the Status to Refunded if the refund is done via PayPal (see separate section about Refunding).

4.6.4. Test the olr-config.php file

During testing, it is convenient to turn on displaying of PHP errors to the user, as described in the section **Customize Security and Error Options** above.

To test the olr-config.php file for PHP syntax errors, simply browse to that file. It should display a blank page, and there should be no errors (displayed or in the error log file).

The PHP system writes errors to a log file (specified by the $c_error_logfile parameter in the olr-config.php file) in the web site folder. Use the web server's file browser to look at this file, or use FTP to copy it to your computer to look at it. If you have displaying of errors turned on, the errors will be displayed in the web page, so you don't need to look at the error log file.

The data about each error includes the date and time, some sort of brief description of the error, and the name of the file and the line number in the file where the error occurred. Go to the indicated line (or sometimes the previous line) and look for a PHP syntax error, like extra or missing quotes, commas, or parentheses, or a missing semicolon at the end of the line.

4.7. Build the System

After the database is created and the olr-config.php file is customized and tested, use your web browser to go to the "olr-build-all.php" file in the web site. In a WordPress web site, the URL will be something like: http://www.yourdomain.com/wp/wp_content/your-event/olr-build-all.php

The page is protected by a login. The login password is specified by the $c_build_reg_login_pwd parameter in the olr-config.php file. The password can be given to whoever might need to build the registration system. The password is remembered while the browser is open, so it only has to be entered once during a browsing session.

In the "Build Database" section, click the "BUILD" button to create the database table for the event.
CAUTION: if the table already exists, it will be deleted and re-created.
A success message should be displayed.
It an error is displayed (or logged in the error log) it is probably caused by an incorrect database name, database user name, database user password, or database table name in the olr-config.php file.

A copy of the SQL that creates the table is saved in the file name specified in the "$c_db_table_def_file" parameter in the olr-config.php file (with the file extension .php added if that is not already present). The file is for reference only. You do not need to do anything with it.

In the "Build Online Registration HTML Page" section, click the "BUILD" button to create the registration HTML page for the event.
The file will be named according to the value of the "$c_online_reg_fname" parameter in the olr-config.php file.
CAUTION: if the file already exists, it will be deleted and re-created.
A success message should be displayed. A link to the file is displayed. Clicking the link will display the registration page in a new browser tab/window.

If the $c_return_url value in the config file is not blank, the Online Registration Page Builder also creates a new file named "online-reg-confirm.html", if it does not already exist. Since the file is not overwritten, any customizations you make are preserved.

The Online Registration Page Builder also creates a new file named "online-reg-closed.html", if it does not already exist. Since the file is not overwritten, any customizations you make are preserved.

Both build functions delete the existing registration and error log files defined in the olr-config.php file: $c_pp_logfile, $c_reg_logfile, $c_error_logfile

If you find missing or incorrect fields in the registration HMTL page, go back to the olr-config.php file and check the field definitions. After making any changes to the olr-config.php file, this build page should be used to re-build the database table and the HTML page.

INSPECT THE REGISISTRATION PAGE CAREFULLY TO MAKE SURE ALL THE DATA FIELDS ARE CORRECT. IT IS NOT PRACTICAL TO CHANGE THE FIELDS AFTER REGISTRATION HAS STARTED.

4.8. Customize Registration HTML Page

The generated HTML page is fully functional and will work fine, but it is not very pretty. The HTML page generator is not very smart. It tries to do some grouping of the fields, but mostly it generates one line for each data field. The Phone, Radio, and Select fields are placed below the normal data fields. You will probably want to change the grouping, position, and order of the generated fields, and change and add some text. You will probably also want to customize the page to match the layout of the rest of your web site.

You may move the HTML elements and re-format the layout however you want, except **DO NOT CHANGE THE TYPE, ATTRIBUTES, OR CONTENT OF THE HTML ELEMENTS <INPUT>, <TEXTAREA>, <SELECT>, <OPTION>, OR <SPAN>**. Doing so will break the system.

Your web site probably has some sort of template used for most of the pages (common header, footer, background color, fonts, etc.). How to integrate the registration page into your web site depends on how your site is constructed. Below are instructions for how to integrate the page into two kinds of web sites: raw HTML/CSS, and WordPress. If your site is constructed some other way, you will have to figure out how to do it yourself, based on the instructions below.

4.8.1. HTML Web Site

You have two options to incorporate the online registration page into your web site: copy the contents of the registration page into a blank web page that uses your template, OR copy the header and/or footer HTML from your web site template into the registration html file.

To copy the contents of the online registration html page into a blank template page:

- Copy a blank template page into the online registration folder, and name it online-reg.html (or whatever you want the registration page to be named).
- Copy the <link rel="stylesheet" …> line from the top of the registration html file into the <head> section of the new page
- Copy the entire tag and its contents from the registration html file into the new page as the content of the body of the new page
- Update the <title> of the page (and the <meta> description tag, if desired).

4.8.2. WordPress Web Site

There are two ways to integrate the online registration page into a WordPress web site: keep the page as a HTML page, outside of WordPress, or copy the contents of the registration page into a new WordPress page.

4.8.2.1. HTML Registration Page

Keeping the registration page in HTML format is the easiest way to incorporate the page into a WordPress web site, but the page will NOT use the WordPress theme, so it will not have the same layout, menu, colors, etc. as the rest of the web site.

You will need to create a link to the registration page in the normal WordPress page about your event. The URL will be similar to:

http://www.yourdomain.com/wp/wp_content/your-event/online-reg.html

You can create a menu item for the registration page by specifying a "Custom Link" and using the full URL to the page, as above.

In the registration page, you should add links to the normal WordPress page about the event, and/or to the web site home page, so people can go back easily.

4.8.2.2. WordPress Page

Incorporating the contents of the registration page into a WordPress page with your theme is more complicated.

- In the registration HTML page, change the "action=" attribute of the <form> tag to add the WordPress directory: action='/wp/wp_content/your-event/olr-pp-send.php'
- In the registration HTML page, near the bottom of the file, change the "src=" attribute of the <script> tag to add the WordPress directory: src='/wp/wp-content/your-event/olr-validate.js'
- In WordPress, install and activate the following two Plugins, if you don't already have them: "Preserved HTML Editor Markup Plus", and "WP Add Custom CSS"
- In WordPress, create a new Page, and open the Page Editor for it.
- Select the "Text" editing tag (NOT the "Visual" tab).
- Copy the entire tag and its contents in the registration HTML page and Paste it into the text editor as the contents of the page.
- Locate the olr-styles-wp.css file in the registration HTML page folder. Open it in any text editor and Copy its entire contents.
- In the WordPress Page Editor, scroll down to the "Custom CSS" section. Paste the contents of the olr-styles-wp.css file into the Custom CSS text box.
- Save/Update the page

4.9. Customize Registration Confirm HTML Page

If the "$c_return_url" item in the config file is used (not blank), the registration page builder will create a new file named "online-reg-confirm.html", if the file does not already exist. The file contains the simple message that the registration was successful. You may customize this file however you want. Once the file already exists, the registration page builder will NOT overwrite it, so your customizations are preserved.

4.10. Customize Registration Closed HTML Page

If the "$c_reg_shutdown_date" item in the config file is used (not blank), the registration page builder will create a new file named "online-reg-closed.html", if the file does not already exist. The file contains the simple message that registration is closed. On and after the shutdown date, anyone trying to access the normal registration page will be re-directed to this page. You may customize this file however you want. Once the file already exists, the registration page builder will NOT overwrite it, so your customizations are preserved.

4.11. Customize CSS Style File

The online registration system has a CSS Style file to define the "look" of the page. It defines simple things like fonts, and adds some fancy formatting like drop shadows on the fieldset boxes, etc. Two CSS files are provided. The generated HTML registration page uses the CSS file "olr-styles.css", which has relatively simple style definitions. WordPress themes typically contain complex styling that conflicts with and overrides the simple definitions in the "olr-styles.css" file. The CSS file "olr-styles-wp.css" contains more complex styles designed to override the WordPress theme definitions. The comments below about customizing the styling apply to both HTML and WordPress web sites. For HTML sites, the customizations are done directly in the "olr-styles.css" file. For WordPress sites, the customizations are done in the "Custom CSS" section of the Page Editor for the WordPress registration page (which was filled in from the "olr-styles-wp.css" file).

Your web site probably has one or more of its own CSS files to define the "look" of your entire web site. Feel free to modify the olr-styles.css file to coordinate with your site CSS file(s). Possible modifications include:

- Remove or disable the "body" definition so the one in your site CSS file is used, or modify it to match the rest of the pages in your web site. An easy way to disable the body (or any other style definition) is to prefix it with "zz" or something similar so it will no longer match a HTML tag or ID (e.g. zzbody).
- Modify the #olr-content definition to change the width, border, shadow, etc.
- The registration page uses <fieldset> to divide the page into sections. If you do not like that, you can remove the <fieldset> tags from the HTML page and format the sections and fields in some other way. In this case, the "fieldset" and "legend" style definitions are not needed.

The registration page uses the following definitions for specific reasons, so they should not be removed from the CSS file, and their definitions should not be changed:

```
#olr-content .olr-label {line-height: 200%;} /* extra vertical space for labels */
#olr-content .olr-err {border: 3px solid red;} /* red box around fields with errors */
#olr-content .olr-line-ht-150{line-height: 150%;} /* extra line spacing */
#olr-content .olr-bold {font-weight: bold;}
#olr-content .olr-required {font-weight: bold;}
#olr-content .olr-left {text-align: left;}
#olr-content .olr-right {text-align: right;}
#olr-content .olr-center {text-align: center;}
#olr-content .olr-top {vertical-align: right;}
#olr-content .olr-bottom {vertical-align: right;}
#olr-content .olr-middle {vertical-align: middle;}
```

When the registration page is implemented as a WordPress Page, the WordPress Theme styling might conflict with the registration page CSS, causing unexpected and undesired formatting. You will have to experiment with the style definitions in the "Custom CSS" section of the Page Editor to resolve any problems.

All the modern browsers have a "Developer" tool which shows exactly which CSS rules are applied to any item in the page, including rules that are overridden by others (indicated by strike through lines). If a WordPress Theme style is doing something you don't like, add or modify a rule in the Custom CSS to override the Theme style.

5. Testing a Registration Page

A new registration page should be tested thoroughly before putting it into production (test all options, and some typical combinations of options).
The system sends some error message to the webmaster, and it displays others to the user and/or logs them to the error log file.

For testing, set the $c_test_mode parameter near the top of the olr-config.php file to 1 to use the PayPal Developer Sandbox instead of the production PayPal system.
Set the $c_email_pending parameter to 1 to send pending emails to the webmaster.
Set the $c_webmaster_email parameter to your email address.
Set the $c_registrar_email for the Sandbox to your email address.

With these settings, error emails will be sent to you.

The data is saved in the database and the Registration Pending email is sent to the webmaster before going to PayPal, so you do not need to complete the transaction in PayPal to test most of the system. Verify the data in the email is correct. You can use the "olr-print-regs.php" and "olr-edit-regs.php" pages to see the pending registration to verify the data is correct (or not).

If the "Pay" button in the HTML registration page goes to a blank screen, that generally indicates an error in the PHP code. Check the error log file. If the error is not in the olr-config.php file, there is either a bug in the system, or data in the olr-config.php file is incorrect or inconsistent enough to cause a problem elsewhere. You might be able to look at the line of code specified in the error to try to figure out what data field was being processed, then check the olr-config.php file for that data field. An alternative is to email the error to the online registration system developer for troubleshooting.

If the "Pay" button in the HTML registration page goes to a new page which says an error has been encountered and the webmaster has been notified, check the email account defined in the olr-config.php file for the webmaster. The system has detected some sort of data inconsistency error, and has sent information about the error to the webmaster in an email. Read the email and try to determine the cause of the error. The error can come from data verification in the registration system, or from the MySQL database.

If you make any changes to the olr-config.php file, you should use the "olr-build-all.php" page to re-build both the database table and the registration HTML page.

If the system seems to be working correctly with no errors, still look in the defined in the config file variable $c_error_log file in the web site folder to verify there are no non-fatal errors that might cause issues later. If the file is not there, then no errors have been found.

If a PHP error is in a file other than the olr-config.php file and you are SURE the olr-config.php file is correct, there might be a problem in the system itself. Please report the problem to the developer of the system via a bug report in the web site where you downloaded the system, or send email directly to the developer.

Testing using the PayPal Sandbox and not completing the transaction should be sufficient to verify your olr-config.php file is correct by verifying the Pending registrations.
If you want to complete the PayPal transaction in the Sandbox, you have two options.
You can email the developer and get the login information for his Sandbox account.

Alternately, you can go to the PayPal Developer web site https://developer.paypal.com/ and create your own account for Sandbox testing.

If the system goes to PayPal and the PayPal payment is completed, but the registration does not change from the Pending to the Paid state and no confirmation emails are sent, check the value of the "$c_notify_url" variable in the olr-config.php file to make sure it is correct. It should be the full URL to the olr-pp-ipn.php file in your web site. Note that sometimes there is a delay in the PayPal processing. It says payment is completed, but it does not send the email back to the registration system for several minutes (sometimes as much an hour).

When testing using the PayPal Developer Sandbox is complete, set the $c_test_mode parameter to 0 to use the production PayPal system. Also, disable display of PHP error messages. Use the olr-build-all.php page to re-build the database to clear all test data and start with a clean empty database. Either you yourself or someone from your organization should make the first registration, and you should verify it worked properly.

PayPal does allow purchases (i.e., registrations) to be refunded (within 60 days), so you could do some final testing using the production web site, and just issue refunds for the test registrations. You can use the olr-build-all.php page to re-build the database to clear all the test registrations and start with a clean database.

There is one quirk about using the PayPal Developer Sandbox. It does not use the non-profit discount fee. The "Pending" registration email is sent BEFORE going to PayPal, so contains the data calculated by the registration system for the totals, including the PayPal fee. The "Paid" confirmation emails are sent AFTER PayPal payment completion, and contain the actual amounts reported by PayPal, and the database is updated with these amounts. If you are using the PayPal non-profit fees in the system, the PayPal Fee in the "Pending" and "Paid" emails will not match when using the developer sandbox, because the "Pending" message will use the non-profit fee, and the "Paid" message will use the regular fee. If you see this mismatch, don't worry. It normal for this situation.

6. Changing the Early or Late Registration Date

After registration has started, you might decide to change the cutoff date for the early/late registration discount/fee, like extending the early registration for a few more days. This can easily be done by changing the date in two files:

- olr-config.php file: change the date value of the option $c_early_late_reg_cutoff_date.
- In the Javascript at the end of the online registration HTML page, change the date in the line containing either/or
  CheckEarlyRegDate('MMMM DD, YYYY', 0);
  CheckLateRegDate('MMMM DD, YYYY', 0);
  where MMMM DD, YYYY is the desired date, in the format 'May 5, 2014'

The date is the last date of the early discount, or the first date of the late fee. For example, setting the date to May 15 means the early registration discount will be applied through the end of May 15, or the late registration fee will be applied on or after May 15.
NOTE: Changing this only works to change the date of an option that was already defined (Early or Late registration). This cannot be used to add Early or Late registration to an event that was not originally configured that way. It also cannot be used to change from Early to Late (or vice versa), or to disable Early or Late registration.

7. Closing Registration

At some point, you probably want to shut down the online registration. There are two ways to do this. The system has an optional feature to shutdown registration automatically on a specified date. To use this, fill in the desired shutdown date in the "$c_reg_shutdown_date" item of the config file. When the registration page is built, an additional file named "online-reg-closed.html" will be created (if it does not already exist). You may customize this file as desired. On and after the specified date, the online registration page will automatically re-direct to the "online-reg-closed.html" page. If you need to change the shutdown date after the system is in use, simply change the date in this Javascript near the top of the online registration file:

```
<script>
CheckShutdownDate('February 22, 2019', 0);
</script>
```

If you did not originally set the date in the "$c_reg_shutdown_date" item in the config file, you can enable the automatic shutdown feature by adding the above Javascript with the desired date to the online registration page, right before the "</head>" tag.

If you prefer to shutdown registration manually, do this:

- Rename the online-reg.html file to an unusual name like zz-closed-online-reg.html so it cannot be easily accessed. In WordPress, change the name of the page and its "slug".
- Create a new online-reg.html file whose contents say the registration is now closed. In WordPress, create a new Page with the name and "slug" of the old page. If you have the registration page in a Menu, you might need to update the Menu item to point to the new page. Alternately, you can keep the same page, and just replace the contents.

Whether the shutdown is done automatically or manually, it is best to disable the files that communicate with PayPal, to prevent accidental or malicious communications. To do this, rename the following files so they are no longer easily accessible: olr-pp-ipn.php and olr-pp-send.php (e.g. zz-closed-olr-pp-ipn.php, zz-closed-olr-pp-send.php).

8. Viewing and Downloading Registrations

The file "olr-print-regs.php" displays the current registrations, with a separate table for each Status (Pending, Paid, etc.), and has a button to download the data in Excel CSV format. The page is protected by a login. The login password is specified by the $c_print_reg_login_pwd parameter in the olr-config.php file. The password can be given to whoever might need to view and download the

registrations. The password is remembered while the browser is open, so it only has to be entered once during a browsing session.

Some people might have made several attempts to register, so might have one or more Pending registrations in addition to a Paid one. The Registration Editor page (see separate section below) can be used to delete the Pending registrations.

If there are Pending registrations for one or more persons with no corresponding Paid registration, it would be a good idea to contact those people and see if they had a problem with the registration process, and help them get registered. If they had payment problems in PayPal, you can ask them to send a check, and their Pending registration can be treated as a paper registration, and converted to a Paid registration using the instructions in the section below about Paper Registrations.

The file "olr-show-names.php" displays just the names of the current Paid registrations, with the total count. There is NO login for this page, so it can be viewed by anyone. The system does NOT automatically create a link to this page. If you want to allow anyone to see who has already registered, you can put a link to this page on your event main page, or wherever you want.

9. Refunding Registrations

If a registered user cannot attend the event and needs a refund, if they paid using PayPal (either with their PayPal account or with a credit card), the PayPal web site can be used to issue a refund, within 60 days of payment. After 60 days, the refund will have to be issued via check instead of via PayPal.

If a refund is issued via PayPal, the system will automatically change the Status of the registration from Paid to Refunded. PayPal allows a partial refund (e.g. refund for one person of multi-person registration), but the online registration system is not smart enough to handle that. The entire registration will be marked as Refunded. To properly handle a partial refund, the Registration Editor can be used to edit the Refunded registration by removing one or more people, changing the payment amounts, and changing the Status (see the section below about Editing Registrations).

If the refund is issued via check, the Registration Editor can be used to manually change the Status of the registration (see the section below about Editing Registrations).

After a registration Status is changed to Refunded, the system does allow a new registration by the same person, but it is treated as a separate NEW registration, not related to the Refunded registration.

10. Editing Registrations

The file "olr-edit-regs.php" allows editing of the registration data. The page is protected by a login. The login password is specified by the $c_edit_reg_login_pwd parameter in the olr-config.php file. The password can be given to whoever might need to edit a registration. The password is remembered while the browser is open, so it only has to be entered once during a browsing session.

After login, the first screen shows a list of all the registrations, with an "Edit" button and a "Delete" button for each one. Click the "Edit" button for the desired registration. This will display all the registration data, and allow most of the fields to be edited.

Each database table field is displayed on one line, with the name of the field followed by its value. The field names are the same as the "name" values in the olr-config.php file, with the data type prefix (e.g. config name = "city", database field = "txt_city").
The data type prefix will help you understand the type of the data to enter valid values.
The "chk_" prefix indicates a checkbox. In the database, the values are stored as 0 and 1.

The registration Status is represented by a selection list containing the valid values, including any custom values specified in the $c_custom_statuses item in the config file.

For the person data, there is a set of fields for each person, from person 1 to person N. The field names have a suffix of the person number, from 1 to N. Examples:
txt_firstname1, txt_lastname1 = first and last name of person 1
chk_child1, txt_childage1, = Child checkbox and Child Age text box for person 1
chk_student1 = Student Checkbox for person 1
…
txt_firstnameN, txt_lastnameN = first and last name of person N
chk_childN, txt_childageN, = Child checkbox and Child Age text box for person N
chk_studentN = Student Checkbox for person N

For musician options, there is a set of fields for each musician, from 1 to N. The field names have "_mus_" in the names, and a suffix for the musician number. The first field for each musician is "txt_mus_nameN", which is the first name of the musician. If you edit the first name of a person in the person data, be sure to make the same change in the musician data (if they are listed there).

For caller options, there is a set of fields for each caller, from 1 to N. The field names have "_call_" in the names, and a suffix for the caller number. The first field for each caller is "txt_call_nameN", which is the first name of the caller. If you edit the first name of a person in the person data, be sure to make the same change in the caller data (if they are listed there).

For the Attendance Options, there is a set of 3 fields for each option, from option 1 to option N. The field names have a suffix of the option number, from 1 to N. Examples:
txt_aname1 = Name of Attendance Option 1
num_aprice1 = Price of Attendance Option 1
num_aqty1 = Quantity of Attendance Option 1
…
txt_anameN = Name of Attendance Option N
num_apriceN = Price of Attendance Option N
num_aqtyN = Quantity of Attendance Option N

The Merchandise Options are similar to the Attendance.Options. Examples:

txt_mname1 = Name of Merchandise Option 1
num_mprice1 = Price of Merchandise Option 1
num_mqty1 = Quantity of Merchandise Option 1

For the Attendance and Merchandise options, you only want to change the "xxxqtyN" values. Do NOT change the "xxxnameN" or "xxxpriceN" values.

When the data changes are complete, click the "SAVE" button to save the changes.
To exit the edit WITHOUT making any changes, click the "Cancel" button.

Registration changes made via the editor do NOT automatically send any emails, and are not logged anywhere. If anyone needs to be made aware of changes to a registration, it is the responsibility of the person making the edits to notify the appropriate people.

11. Deleting Registrations

The file "olr-edit-regs.php" allows deleting any registration in addition to editing it (see above).
After login, the first screen shows a list of all the registrations, with an "Edit" button and a "Delete" button for each one. Click the "Delete" button for the desired registration. This will pop up a confirmation prompt. Click "OK" to confirm the deletion.

12. Importing Paper Registrations

Most people use an online registration system when it is available, but a few still use paper registration forms. It would be nice if ALL the registrations were listed in one place. With a little extra effort, it is possible to add paper registrations to the online registration system, so it holds the data for ALL registrations. This works best when the configuration parameter $c_email_pending = 1, so an email is sent to the webmaster for all Pending registrations. Here is how to add paper registrations to the online registration system:

- Registrar receives a paper registration.
- Registrar uses the online registration page to enter the data in the paper registration. Be sure to NOT check the box to include the PayPal fee. Click the Pay button, but do NOT complete the PayPal payment. Just close the PayPal payment page.
- If the Registrar does not have the password to use the Edit Registration page, s/he sends email to the webmaster to indicate that the Pending registration is for a paper registration, and requests it be changed to Paid.
- The Registrar or the webmaster uses the Edit Registration page (olr-edit-regs.php) to change the Pending registration to Paid, by changing the Status field from Pending to Paid and by filling in the num_totalnet field with the same value as num_totalgross.
- If the user requested or offered hospitality, the webmaster forwards the Pending registration email to the Hospitality coordinator, explaining that it was a paper registration. Change the email text to indicate it is Paid instead of Pending.
- Optional: the webmaster can forward the Pending registration email to the user to confirm registration, changing the email text to indicate it is Paid instead of Pending.

Once in the system, the Paid online and paper registrations can be distinguished by the TransactionID field. This field is filled in by PayPal. For paper registrations, it will be blank.

13. Robot Web Site Scans

Google and other "robots" periodically scan web sites to extract data for search indexing, etc. Also, some hackers scan web sites for malicious purposes. A scan of a php file like olr-pp-send.php will not contain the proper data fields, so will trigger an error message to the webmaster for missing or incorrect data. Since the system reports an error instead of trying to process the data, no harm is done to the system.

Scans by "robots" can theoretically be reduced by using a file named "robots.txt" in the top level folder of the web site, with instructions for which files to ignore. To tell robots to ignore php files, use this line:
Disallow: /*.php$

An example of a robots.txt file is included in the Examples folder. The robots.txt file should be located in the top level folder of the web site, not in the online registration system folder.

Note that "robots" are NOT required to read and honor the contents of robots.txt files. Major search engines like Google probably do read and honor the robots.txt files, but malicious scanners probably ignore robots.txt files. Use Google or some other search engine to search for "robots.txt" to find more information about robots.txt files.

An application named SiteLockSpider can be used to scan a web site to find vulnerabilities in or improvements to web site files. This is normally a paid service, but some web hosts are supplying a periodic automated free scan, which cannot be disabled. The online registration system has been updated to ignore scans by SiteLockSpider, so no error messages are sent for those scans.

14. Error Messages

There are two levels of error messages in the system: PHP script errors, and data verification errors.

The most common PHP errors are syntax errors in the olr-config.php file. These must be corrected before they system will work, so there should be no PHP errors during normal operation. It is possible that some strange combination of data entered by the user could cause a PHP error, but that is very unlikely. With display of PHP errors turned off during production, errors are only written to a log file, with no notification to the webmaster. It is recommended to check the error log file every week or two.

The system does extensive validation of the data submitted by the user. Some simple data errors are reported to the user, so the error can be corrected and the registration can be re-submitted. Duplicate registrations are also reported to the user.

Other more serious errors are reported to the webmaster via email. The email contains a description of the error, a dump of all the data fields submitted by the user (POST data), and a dump of all the Server/Header data.

The Server/Header data field HTTP_USER_AGENT indicates the type of the "browser" used to access the system. A value indicating Google or Yahoo or some other search engine indicates a benign scan for search indexing purposes. Other values like "Java/1.6.0_04" might indicate a malicious scan.

The Server/Header data field REMOTE_ADDR contains the IP address of the "browser". The internet domain tool "Whois" can be used to determine the approximate location of this IP address, and the internet service provider.

15. Data Logging

The system maintains several data log files, which can be useful for various troubleshooting purposes. These files are created automatically in the web site folder, and need no management. For security purposes, the .htaccess file included with the system prevents accessing files named *bak, *csv, *ini, *log and *sql via the web browser. To view the files, either login to the web server account and use the file manager provided by the web server, or use FTP to copy the files from the web server to your computer.

- PHP error log – a text file that contains errors in the PHP code, managed by the PHP system. The actual filename is **listed** by the $c_error_logfile parameter in the olr-config.php file. It is **defined** by the .user.ini or the .htaccess file in the online registration folder. See the section **Customize Security and Error Options** above.
- registration log – a text file in Excel CSV format which contains all the registration data (both pending and paid). The column names are the same as the database table columns, so this could be considered a backup to the database, and could be used to re-create the database if the database failed for some reason. The actual filename is specified by the $c_reg_logfile parameter in the olr-config.php file.
- paypal log – a text file contains all the payment information messages sent from PayPal to the system, and a log of what the system did with each one (emails sent, etc.). This might be useful for tracking down problems with a specific registration. The actual filename is specified by the $c_pp_logfile parameter in the olr-config.php file

16. Resources

Information about and a demo of the registration system can be found at:
https://www.glennman.com/online-reg/home.html

The system can be downloaded from: https://sourceforge.net/projects/gmonlineregistration/

For more information and help, contact the developer:
Glenn Manuel gemdancer@fastem.com 972-963-0045 (Dallas, TX)