******

***Information Technology Summary***

***And Operational Procedures***

***Rev 2.2***

**Overview**

This document outlines various systems, applications, scripts and other Information Technology employed in support of the operation of Gay Naturists International, Inc. This Rev 2.x is a major revision to reflect the migration to the Internet4Associations (I4A) member management DB/app.

The following systems/applications are documented:

[GNIAdmin 3](#_Toc70691415)

[GNI Event Management Utilities 3](#_Toc70691416)

[GNIAdmin: Uploads & Updates 4](#_Toc70691417)

[Upload Registration Confirmation Info & Generate Paperless Confirmation Emails 4](#_Toc70691418)

[Annual Maintenance Activities for Upload Registration Confirmation Info & Generate Paperless Confirmation Emails 4](#_Toc70691419)

[Cancel A Registration 5](#_Toc70691420)

[Upload Cabin Assignments/InOut Data 5](#_Toc70691421)

[Turn OFF/ON Auto Roster Update From I4A 5](#_Toc70691422)

[Force Roster DB Update From I4A 6](#_Toc70691423)

[Mark All In/Out Attendees As OUT 6](#_Toc70691424)

[Update I4A Reference Table 6](#_Toc70691425)

[GNIAdmin: Configuration & Utilities 6](#_Toc70691426)

[Config For Registration/Cabin Assignment Upload 6](#_Toc70691427)

[Configuration for Registration Confirmation Email 7](#_Toc70691428)

[Configuration for Registration Confirmation Email Annual Maintenance Activities 9](#_Toc70691429)

[Configuration for SmartWaiver Interface 9](#_Toc70691430)

[Change Auto Roster Update Frequency (Every xxx Minutes) 9](#_Toc70691431)

[Edit Roster DB Update Settings 10](#_Toc70691432)

[EventMobi/Print Kiosk Configuration 10](#_Toc70691433)

[Lookup Member Info 10](#_Toc70691434)

[GNIAdmin: Utility Procedures 10](#_Toc70691435)

[Change Password 10](#_Toc70691436)

[Provides the currently-logged in operator to change their password. 11](#_Toc70691437)

[Manage User IDs 11](#_Toc70691438)

[System 11](#_Toc70691439)

[InOut Check In/Attendance Tracking/Smartwaiver 12](#_Toc70691440)

[Smartwaiver 13](#_Toc70691441)

[Annual Maintenance Activities 14](#_Toc70691442)

[Paperless Registration/GNI-I4A Interface 15](#_Toc70691443)

[The Process 18](#_Toc70691444)

[Annual Maintenance Activities 19](#_Toc70691445)

[Print Kiosk: Overview 20](#_Toc70691446)

[**https://legacycollector.org/firefox-addons/1659/icon.pngR-kiosk** 20](#_Toc70691447)

[Print Kiosk: EventMobi 21](#_Toc70691448)

[Camp Map 22](#_Toc70691449)

[Attendance Roster Button 22](#_Toc70691450)

[Admin Module/Configuration File 22](#_Toc70691451)

[Annual Maintenance Activities 23](#_Toc70691452)

[Print Kiosk: Attendance Roster 24](#_Toc70691453)

[Roster Database Generation 25](#_Toc70691454)

[Reporting 25](#_Toc70691455)

[Annual Maintenance Activities 25](#_Toc70691456)

[Summary of Annual Maintenance/Update Procedures 26](#_Toc70691457)

[Paperless Registration Confirmation Annual Maintenance Activities 26](#_Toc70691458)

[Configuration for Registration Confirmation Email Annual Maintenance Activities 26](#_Toc70691459)

[Smartwaiver Annual Maintenance Activities 26](#_Toc70691460)

[Print Kiosk: Eventmobi/Schedule Annual Maintenance Activities 26](#_Toc70691461)

[Print Kiosk: Attendance Roster Annual Maintenance Activities 26](#_Toc70691462)

[IDs/Passwords 28](#_Toc70691463)

As of this writing, the systems/apps (source code and operational environment) listed above reside on a GoDaddy server under the <https://www.>clubpittsburgh.com domain. The public-facing GNI website is housed at Internet4Associations (**I4A**) ([www.internet4associations.com](http://www.internet4associations.com)). (This document does not address the public GNI website.) It is expected that these GNI utilities will migrate to a GNI-owned or controlled server. The only requirements for the target hosting site is PHP 5.0 or later.

Throughout this document, when referencing one of these apps/subsystems, the host domain prefix <https://www.clubpittsburgh.com> will be written “**~host”** as shorthand (e.g., **~host/inout**).

These subsystems are all custom code built using PHP (currently Version 5.6.40) on the server side and Javascript (JS)/HTML/CSS on the client side. The code is “scratch built” in that no formal framework has been employed. This developer uses WinSCP, an FTP client on Windows, to access the source code.

Each system is configurable through a purpose-built administration program, as described in the corresponding sections.

These systems have been built piecemeal over a number of years. As a result, and due to added or changing requirements and other issues, some may be more disjointed than others. The nature of the underlying data is also a product of a multi-decade evolution that has created some data structure horrors. These modules attempt to clean up those issues where possible and hack around them where needed. The code is reasonably well documented (well, at least in my opinion; Lol) and should be maintainable by any qualified software engineer. This document is intended to provide a road map and overview of the various systems and their operation.

# GNIAdmin

## GNI Event Management Utilities

The *GNIAdmin* app, also known as the Event Management Tools, is a collection of utilities and data managers that facilitate various aspects of registration and operation of GNI events. It also provides a “link” between GNI member’s I4A profiles and many of the functions outlined in this document. (I4A provides administration access to GNI’s member data at [**https://www.gaynaturists.org**](https://www.gaynaturists.org); login criteria can be found in the *IDs/Passwords* section of this document).

The utility is accessed at **~host/gniadmin**. The app is comprised of the following scripts found in that directory:

|  |  |  |
| --- | --- | --- |
| CPCE\_check\_session.php.inc | GNIREG\_link.css | GNIREG\_link\_load\_excel.html.inc |
| CPCE\_classes.php.inc | GNIREG\_link\_cancellation.html.inc | GNIREG\_link\_load\_excel.js |
| CPCE\_edit\_ids.html.inc | GNIREG\_link\_cancellation.js | GNIREG\_link\_load\_excel.php |
| CPCE\_edit\_ids.js | GNIREG\_link\_cancellation.php | GNIREG\_link\_load\_lib.php.inc |
| CPCE\_edit\_ids.php | GNIREG\_link\_config.html.inc | GNIREG\_link\_main.html.inc |
| CPCE\_edit\_password.html.inc | GNIREG\_link\_config.js | GNIREG\_link\_main.js |
| CPCE\_edit\_password.php | GNIREG\_link\_config.php | GNIREG\_link\_main.php |
| CPCE\_footer.html.inc | GNIREG\_link\_email.php.inc | GNIREG\_link\_user\_info.html.inc |
| CPCE\_globals.php.inc | GNIREG\_link\_email\_config.html.inc | GNIREG\_link\_user\_info.js |
| CPCE\_header.html.inc | GNIREG\_link\_email\_config.js | GNIREG\_link\_user\_info.php |
| CPCE\_lib.php.inc | GNIREG\_link\_email\_config.php | GNIREG\_link\_waiver\_config.html.inc |
| CPCE\_login.html.inc | GNIREG\_link\_foot.html.inc | GNIREG\_link\_waiver\_config.js |
| CPCE\_style.css | GNIREG\_link\_head.html.inc | GNIREG\_link\_waiver\_config.php |
| CPCE\_system.php | GNIREG\_link\_inout\_utils.html.inc | I4A custom GNI API.pdf |
| CPCE\_system\_ver.html.inc | GNIREG\_link\_inout\_utils.php | index.php |
| CPCE\_system\_ver.php | GNIREG\_link\_lib.js | phpinfo.php |
| CPCE\_user\_lookup.php.inc | GNIREG\_link\_lib.php.inc | PJKUTIL\_lib.js |
| GNIREG\_globals.php.inc | GNIREG\_link\_load\_cabin\_excel.php |  |

All of these scripts serve a function, but some of their functionality may be “legacy” and no longer needed (although that will not interfere with their operation). For example, the scripts starting with *CPCE\_* (and *phpinfo.php, PJKUTIL\_lib.js*) were “borrowed” from other work developed by P.J. Karlovich and contain some functionality that doesn’t apply here. However, their basic operation of managing User IDs/Passwords (and other functions) is useful (more on that later).

Upon startup, the *GNIAdmin* app initiates an API call to I4A to obtain a dataset that provides a “link” between I4A’s contactID (a unique identifier for each record in their DB) and the GNI-generated MemberKey (which may not be unique in the I4A system, as members sometimes have multiple entries depending on their status, such as Cabin Coordinators). The I4A API documentation can be found at <https://api.i4a.com/index.php/I4A_api_guide>.

This “link” file also contains the email and altEmail addresses on file at I4A for the given member. These are used, along with those from the GNI “offline” DB (a/k/a Rick’s Access DB) when sending out paperless registration confirmation notices.

The I4A API is fairly quick and the data set returned is relatively small. However, upon startup, the module will display an “Updating Local DB…” status while the API request is being serviced, and a few of the options which rely on this data will be unavailable (their buttons will be disabled/greyed out). However, under normal circumstances, this should only take a second or two. (With a previous member database provider, MemberClicks, this was an EXTREMELY lengthy and complex process that sometimes took minutes to complete; this is no longer the case.)

In general, I4A API *retrieval* calls invoke a pre-defined “View” into the I4A DB (they can also reference a DB Table directly). Three custom Views generated for these utilities:

**pjk\_gni\_get\_member.sql**

**pjk\_gni\_get\_roster.sql**

**pjk\_gni\_get\_registration\_baseline.sql** (this is the one invoked upon startup)

The SQL source can be found in **~host/gniadmin/I4A sql**. Each source module is used to create the correspondingly-named View in the I4A Tools utility. I4A Tools can be accessed at [**https://www.gaynaturists.org**](https://www.gaynaturists.org) using the Admin ID/Pwd found in the *IDs/Passwords* section of this document. To change a View, edit the SQL source, then navigate to the **Report Builder** functions in I4A Tools and apply the updated source to the appropriate entry. Information about I4A DB tables can be found in the **Table Information** function. Both of these options are found under the **Tools** menu item.

Other than the “link” process, the *GNIAdmin* module provides options to initiate certain processes or update certain configuration and data fields. Below is a description of each available option.

## GNIAdmin: Uploads & Updates

### Upload Registration Confirmation Info & Generate Paperless Confirmation Emails

The purpose of this function is to update the I4A profile of a member to indicate attendance at an event and also email a registration confirmation message to the member.

At present, new fields are added to I4A profiles FOR EACH EVENT. For example, in 2021, a GNI I4A profile contains the following fields (or “columns” in I4A parlance): ***c\_user\_2017Gathering, c\_user\_2018Gathering, c\_user\_2019Gathering, c\_user\_2021Gathering,*** etc. Each of these fields contains a value of “1” (true or yes) or “0” (false or no) to indicate if the member attended the event that year. There are also fields for ***c\_user\_GNI\_Gathering\_Cabin\_No2017, c\_user\_GNI\_Gathering\_Cabin\_No2018, c\_user\_GNI\_Gathering\_Cabin2019***, etc. which contain a textual cabin number for the corresponding event. These custom fields are currently established through the year **2030**.

This function enables the user to browse to an Excel spreadsheet containing a list of attending members (each identified by unique MemberKey). The spreadsheet must contain column headers identifying the data in each column. These column headers are “mapped” to the corresponding I4A columns based on entries in the ***Current Config*** selected under the “*Config For Registration/Cabin Assignment Upload”* option (see below). The spreadsheet may contain columns that are not needed/referenced by the map; these data will be ignored.

Once the spreadsheet has been identified, the **[Upload]** button causes the *GNIREG\_link\_load\_excel.php* script to take over (documentation internal to that script provides more information on this process) and programmatically updates the I4A profiles through the I4A REST API based on the ***Current Config*** selection. A Registration Confirmation Email will also be generated and send to the member (see also the *Paperless Registration Interface* section found elsewhere in this document).

Note that, based on the number of entries being updated, the speed of the communication line and congestion of the I4A API, this process may run for several minutes. At the conclusion a report will be presented to indicate any warnings or errors that may have been encountered. In most cases, the offending items can simply be corrected and uploaded again.

### Annual Maintenance Activities for Upload Registration Confirmation Info & Generate Paperless Confirmation Emails

* Update the ***Current Config***under the *“Config For Registration/Cabin Assignment Upload”* option appropriately for this year’s event.
* Review/update the confirmation email parameters under the *“Configuration for Registration Confirmation Email”* option as needed.
* Review/update the **Confirmation Email HTML Template** for this year’s event as needed.

### Cancel A Registration

As the name implies, this function will “cancel” or undo the results of the “*Upload Registration Confirmation Info…*” and the “*Upload Cabin Assignments/InOut Data*” functions. It will:

* Remove Registration Data (attendance flag, registration date, cabin number) from the member’s I4A profile
* Remove the Member's entry from the Attendance Roster
* Remove the Member's entry from the InOut Attendance Tracker.

Usage of this function should be self-explanatory. It should be noted that the operator should confirm the Current Configuration selection as shown on this function’s landing page.

A registration can be cancelled by specifying either a MemberKey or member’s last name. Appropriate confirmation messages will be presented before the cancellation is complete.

### Upload Cabin Assignments/InOut Data

Similar to the *Upload Registration Confirmation Info* function described above, this function will update housing assignments to member’s I4A profiles. In addition, it will update the In/Out Tracking database with member data. More information is available (e.g., spreadsheet data fields) on this function’s landing page.

Also similar to the *Upload Registration Confirmation Info* function the operator should verify the Current Configuration map is selected.

Note that, based on the number of entries being updated, the speed of the communication line and congestion of I4A API, this process may run for several minutes. At the conclusion a report will be presented to indicate any warnings or errors that may have been encountered. In most cases, the offending items can simply be corrected and uploaded again.

### Turn OFF/ON Auto Roster Update From I4A

See also the *Print Kiosk: Attendance Roster* section for more information.

When any Attendance Roster page is open in any browser (e.g., the Print Kiosk) a timer causes the Roster to be updated periodically to reflect any changes made by members to their own profile or by GNI operations to the attendance records. This update can be as frequent as 1 minute or as long as desired (see *Change Auto Roster Update Frequency*). This function is used to toggle the automatic update function on and off.

The Roster data is updated through an I4A API call to a View named/configured by **pjk\_gni\_get\_roster**. See also **pjk\_gni\_get\_roster.sql** for more info.

NOTE: If auto update is turned off, the aforementioned *Change Auto Roster Update Frequency* option is not visible. See also the *Print Kiosk: Attendance Roster* section.

### Force Roster DB Update From I4A

This function can be used to cause an immediate update of the Roster database from I4A data. This can be useful if a major or important change has been made to something in I4A and you wish for the change to be immediately visible, without waiting for the auto update to kick in. Simple select this option and the Roster will be updated.

### Mark All In/Out Attendees As OUT

At the end of an event, sometimes not all attendees will have been marked as “Out” in the InOut Attendance Tracker. This function provides a bulk update capability to mark as “Out” any remaining stragglers. The landing page requires 2 pieces of information:

* The date to be used as the check-out date (time currently defaults to 11:00 am; it can be changed in *GNIREG\_link\_inout\_utils.php* script)
* The event year. Note that the InOut process constructs its data file name using the current year. This may create several problems:
  + It won’t allow for *future* events where we may want to set up next year’s event in advance
  + It won’t behave well if the event spans 2 years
  + It won’t (easily) accommodate 2 or more events in the same year (you would need to delete or rename the unwanted data file to permit a new one to be generated)

Should any of these issues create a problem the code will have to be modified to accommodate more unique data file naming.

### Update I4A Reference Table

The GNI/I4A data link is established at startup of these admin utilities and is not refreshed during the current session. However, there may be times the operator wishes to “force” an update of the local reference table (e.g., changes have been made to I4A data and rather than logging out/in to the *Admin* module, this function can simply be invoked).

Once invoked it behaves identical to the startup process of the *GNIAdmin* module.

## GNIAdmin: Configuration & Utilities

### Config For Registration/Cabin Assignment Upload

This function provides an ability to generate a “map” between GNI data fields and I4A DB columns. Multiple maps or "Configuration Sets" can be maintained to facilitate different types of events or for historical context. The most recently accessed/edited set is identified by the "Currently Active Configuration".

The active configuration is the one used to control the Registration Confirmation upload and the Cabin Assignment upload. Each configuration consists of a table which contains a list of map items, each of which contain 3 elements:

|  |  |
| --- | --- |
| ***The Excel Spreadsheet Column Header:*** | This contains the name of a column header in an Excel spreadsheet that contains data to be uploaded to the I4A database, e.g., Nickname. CabinNo, etc. (these header names are NOT case sensitive). Note that, depending on the *Data Type* (see next item) this field may be disabled. |
| ***The Data Type:*** | This identifies or controls the “type” of spreadsheet data being uploaded as follows:   |  |  |  | | --- | --- | --- | | ***Code*** | ***Select List Option*** | ***Description/Usage*** | | T | Text | Data will be uploaded “as is” | | TNB | Text (non-blank) | Data will be uploaded ONLY if it is not blank | | TF | T/F - Use Spreadsheet Value | Spreadsheet values of T, TRUE & 1 will upload as 1; all other values as 0 | | TFT | T/F - Always True | The identified I4A column data will be set to 1 for this member regardless of the spreadsheet value | | TFF | T/F - Always False | The identified I4A column data will be set to 0 for this member regardless of the spreadsheet value | | D | Date - Use Spreadsheet Value | The spreadsheet data will be uploaded after a “format check” to ensure it resolves to MM/DD/YYYY (to the extent possible, the script will reformat the data as necessary) | | NOW | Date - Use Today's Date | The identified I4A column data for this member will be set to today’s date, regardless of the spreadsheet value | | C | Cabin Number | This data will only be uploaded when using the “Upload Cabin Assignments/InOut Data” funcion | |
| ***The I4A DB Field Name:*** | This contains the name of the I4A database column to receive uploaded data, e.g., c\_user\_nickname, c\_user\_2021Gathering, c\_user\_GNI\_Gathering\_Cabin\_2021, etc. (these column names are NOT case sensitive). Note that, at present, only columns that begin with **c\_user\_** are eligible for upload; I4A built a custom interface for GNI to manage these fields (the “public“ I4A API is read-only by default). |

The operator should always review the ***Current Config*** settings before performing any update or delete operations.

### Configuration for Registration Confirmation Email

This function is used to customize and control the format and content of the Registration Confirmation Email that is sent to each member when their registration data is uploaded to I4A (as outlined in the *Upload Registration Confirmation Info & Generate Paperless Confirmation Emails* function above). The email is generated from a template that is stored as an HTML file. The HTML document may be stored anywhere that is convenient; alternatively, this option provides an ability to upload an HTML document to this environment. A sample/baseline of this template can be found at **~host/gniadmin/HTML\_docs/PaperlessConfo.html.txt** (that **.**txt file also contains additional documentation on this process).

The template is a combination of an HTML document with pertinent, relatively static information in HTML format. Embedded throughout are two types of "substitution variables," ones bracketed by **%%GNI%%** (e.g., **%%GNI%%logo%%GNI%%**) and ones bracketed by **%%MEMBER%%** (e.g., **%%MEMBER%%FirstName%%MEMBER%%**).

When an email is triggered to be sent, the **%%GNI%%** fields will be replaced by "**Fixed Substitution Fields**," "**Variable Substitution Fields**" and "**Bus Substitution Fields**" as found in this function and listed under the Tabs across the top of the screen. The **%%MEMBER%%** fields will be replaced with data found on the uploaded confirmation data spreadsheet.

The first Tab of this function contains some control fields, such as the sender’s name (e.g., GNI), sender’s email, and various other fields. Of note is the **Suppress email** field. This identifies a column on the registration data spreadsheet that will suppress the generation of an email to that member (any value in this column will suppress email generation).

The **Fixed Substitution Fields** should be self-explanatory. They represent data that rarely change, such as a link to our logo, and others that are “automatic”, such as the current year. Of note: The Ordinal Year field (e.g., 32nd) is calculated from a base year of 2017, which was GNI’s 32nd year. We have since gotten “out of sync” with that counting due to skipped years, and an “adjustment” attribute is available to correct this. Please see the “*Data-Driven Form Template*” note below.

The **Variable Substitution Fields** Tab provides an ability to specify an endless number of fields and values (URL or email address). The various components available for each variable substitution field are as follows:

|  |  |
| --- | --- |
| Field Name | Your name for this substitution field (optional) |
| Value | Textual data (could be an email address) or a fully-qualified URL to an image or document |
| HTML Substitution Label | The “name” of the substitution label, e.g., **BizEmail** that will be substituted with “Value”.  In other words, any references to **%%GNI%%BizEmail%%GNI%%** will be replaced with the “Value” specified in the field above. An example:  If any of the information above is incorrect, please contact the GNI Business Office at %%GNI%%BizEmail%%GNI%% for assistance.  Another example may be **DrivingDirections** where the Value would be something like <https://somehost.com/gni/documents/Driving_instructions.pdf>. In this case, the HTML email template would look something like this:  <span class="bold">Driving Directions:</span> Driving directions can be  <a href="%%GNI%%DrivingDirections%%GNI%%" target="\_blank">found here.</a>  </span>  When the data is a URL, you can test its accuracy by pressing the “Test URL” button. |
| Your Notes | Notes to yourself to remind you what this field is for. |

Using this mechanism, new documents (e.g., driving directions) can be created and pointed to without having to edit the HTML document. Alternatively, you could hard-code these links into the HTML itself.

The **Bus Substitution Fields** control what bus information, if any, is included in a member’s registration confirmation email. It is expected that there are 2 columns on the Registration Document Upload that correspond to the Arriving and Departing bus for this member. The names of these columns are specified on this Tab. In addition, each bus “route” is composed of the following information:

|  |  |
| --- | --- |
| Bus Code | Any letter/number combination to identify this “route”, e.g., “A” or “A1” |
| Description | The text to be printed in the email where this bus’ substitution code is seen (e.g., **%%GNIBUS%%A1%%GNIBUS%%** |
| Info Sheet URL | A fully-qualified URL pointing to an information sheet about this bus “route”. The member receiving the registration confirmation email can click on this link to obtain information about their scheduled bus transportation. As with the Variable Substitiution Fields, a test button is available to ensure the accuracy of the URL |
| Your Notes | Notes for your future reference as needed |

Again, the HTML template found in **~host/gniadmin/HTML\_docs/Paperless Confo.html.txt** can provide more information.

*Data-Driven Forms Template*

The field names, data descriptions and certain control features that drive this configuration option are taken from an XML document located at **~host/gniadmin/db/GNIREG\_email\_config.xml**. This provides a good deal of flexibility for adding or removing fields. The XML file itself contains some explanation on its use.

Of note is an “adj” attribute on one of the “fixedField” elements. The “ordinalYear” element contains this attribute. It is not required. If present, it may contain a positive or negative integer. This value is applied when calculating the “ordinal year” of an event; counting starts at 2017, which was GNI’s 32nd year. As of this writing, it is set to -2 to account for the 2 years lost to Covid-19…

### Configuration for Registration Confirmation Email Annual Maintenance Activities

* Check/update the HTML email template document if needed
* Check/update the Bus Substitution Fields as needed

### Configuration for SmartWaiver Interface

See also *Smartwaiver* under the *InOut Check In/Attendance Tracking/Smartwaiver* section below.

This function provides the ability to specify the current ***SmartWaiver Template ID*** and ***SmartWaiver API Key*** needed for the Smartwaiver interface of the InOut Attendance Tracker. The API Key is unique to GNI and should never change. The Template ID is unique to each individual template created. To date, we have been using only 1 and the ID has not changed. (The template may be edited as needed without affecting the Template ID).

### Change Auto Roster Update Frequency (Every xxx Minutes)

See also the *Print Kiosk: Attendance Roster* section (below) and the *Auto Roster Update* section (above) for more information.

This function enables the operator to select the frequency that the Attendance Roster is updated with new data from I4A. It can be as frequent as every minute or as infrequent as desired. It is recommended to strike a balance here so as not to overburden the communication line with updates that are too frequent. But also, be cognizant that a member may see some of their information displayed on the roster (e.g., their email address) and wish to hide that information (or even their entire profile). They can do this by modifying the privacy settings in their I4A profile to restrict the data as desired. We would not want them to have to wait an hour until this update is picked up by the roster update process.

The recommended refresh setting is 10 minutes.

### Edit Roster DB Update Settings

This function provides maintenance ability for the 2 data fields that drive the generation and display of the Attendance Roster on the GNI Print Kiosk. Selecting this option will enable editing on the following fields:

|  |  |
| --- | --- |
| **Event Name** | This information is displayed at the top of the Roster display and printed as a header when the Roster is printed using the GNI Print Kiosk. |
| **I4A Cabin Assignment Field** | This is the GNI-specific I4A data column that is examined when gathering the Roster data. The presence of any data in the identified column will cause that member’s record to be included in the Roster.  NOTE: Any restrictions the member has made in their I4A profile to limit visibility of their directory entry or email/phone fields will be respected. |

### EventMobi/Print Kiosk Configuration

This option provides the ability to control the interface with EventMobi (EM) which is used to populate the Event Schedule on the GNI Print Kiosk.

The basic operation is to invoke an EventMobi API and retrieve all available data for the current event, identified by the Event Code (see below). The data is then analyzed and presented on the Print Kiosk screens. The meaning of each of these fields is as follows:

|  |  |
| --- | --- |
| **Kiosk is Currently** | Indicates if the Kiosk is “Enabled” or “Disabled”. A link to the Kiosk pages can be made available on the GNI website if desired. This option enables the Kiosk to be “dark” until ready to go live. |
| **Event Name** | This is the heading that will appear at the top of the Event Schedule display and also on the printed Event Schedule. |
| **Event Code** | This is the EventMobi code for this year’s event. Unfortunately, EventMobi does NOT provide an ability to get this code online. (If you open the “Events List” on the EventMobi Admin page, it does show a column labeled “EVENT CODE”, but this is NOT the code that is used by the API.) To obtain the code for the current event, select “Integrations” from the EM Admin page (see *IDs/Passwords* at the end of this document), then select “Actions” for the GNI entry and then “Email Key”. This will generate an email containing the API Key and Event Codes. |
| **API Key** | This is GNI’s key to access the EventMobi API. Unless the key needs to be regenerated for some reason, there is no reason to change this. |
| **Update Freq (secs)** | This is the frequency with which the Kiosk will query EventMobi for updated schedule information, in seconds. 10 minutes (600 seconds) is recommended. |
| **Camp Map URL** | This is a URL to an image (JPEG, PNG, etc.) of a map to the campground. |

### Lookup Member Info

This provides a quick and easy way to retrieve a member’s information. Its operation should be self explanatory.

## GNIAdmin: Utility Procedures

These utility functions are found at the bottom of the page in the fine print.

Change Password

Provides the currently-logged in operator to change their password.

Manage User IDs

Provides the ability to create, edit or delete user IDs for other operators of the system. This function provides for 3 levels of authority: Standard, Administrator and Super User. The only real restrictions based on user levels is:

* A Standard user cannot create, edit or delete any other user IDs
* An Administrator user can create, edit or delete Standard and Administrator IDs but not Super User IDs
  + Also, an Administrator cannot set any ID’s authority higher than Administrator
* A Super User can do anything (except delete the last ID)

The edit forms provide for the entry of an email address, but this data is not required and is currently not used for anything. (This code was “borrowed” from another project built by P.J. Karlovich.)

### System

This function provides:

* An ability to check the configuration settings (phpinfo) of the current server
  + This can be quite useful if certain commands or functions are not working
* An ability to check version information/modification dates of the *GNIREG* Admin scripts
  + This module is also “borrowed” code from other projects and is of little value here
* An ability to execute a Unix/Linix system command without having an OS login userid
  + In rare instances, can be helpful for debugging

The operation of these functions should be self-explanatory.

# InOut Check In/Attendance Tracking/Smartwaiver

The InOut app provides an ability to track when attendees arrive at or leave a GNI event. In addition, various reports and statistics are available for each day of the event. It is accessed at **~host/inout** and requires an ID/Password to access (the module uses the same IDs/Pwds as the *GNIREG Admin* module). This is a fairly complex process (especially the waiver component); I have tried to heavily document the code as appropriate.

It is driven by an XML data file named “*INOUT**\_names\_db\_YYYY.xml*” where “YYYY” corresponds to the current year (e.g., *INOUT\_names\_db\_2019.xml)*. This data file is stored at **~host/gniadmin/db** and is created by the *GNIAdmin* “Upload Cabin Assignments/InOut Data” function (specifically, the *GNIREG\_link\_load\_cabin\_excel.php* process). This process takes an Excel spreadsheet and updates the cabin assignments in each listed member’s I4A profile (note that generating the InOut data file is merely a byproduct of uploading the cabin assignments; see the *GNIAdmin* module documentation for more information on cabin assignment updating). The Roster data file is automatically and correctly maintained regardless of how many times a cabin assignment is uploaded.

NOTE: As currently implemented, this app will only accommodate one event per year (unless you manually delete or rename the data files to allow new, empty ones to be generated for a subsequent event). It should be a relatively simple matter to modify the code to provide more control of the data file names…

In addition, this process is integrated with a commercial product called Smartwaiver ([www.smartwaiver.com](http://www.smartwaiver.com)). When checking in a member the operator has the option of generating an electronic Waiver of Liability for the member to sign. Smartwaiver provides a good API to enable pre-filling of the waiver with pertinent member information. A visual indicator on the InOut attendee list identifies those members who have not yet signed a waiver.

This module is comprised of the following source files located in **~host/inout**:

|  |  |  |
| --- | --- | --- |
| index.php | INOUT\_lib.php.inc | INOUT\_reports.php.inc |
| INOUT\_check\_session.php.inc | INOUT\_login.html.inc | INOUT\_reports\_disp.html.inc |
| INOUT\_classes.php.inc | INOUT\_login.php.inc | INOUT\_reports\_disp.js |
| INOUT\_edit\_password.html.inc | INOUT\_main.html.inc | INOUT\_reports\_disp\_style.css |
| INOUT\_edit\_password.php | INOUT\_main.js | INOUT\_style.css |
| INOUT\_foot.html.inc | INOUT\_main.php.inc | INOUT\_sw\_api.php.inc |
| INOUT\_globals.php.inc | INOUT\_reports.html.inc | INOUT\_utilities.php.inc |
| INOUT\_head.html.inc | INOUT\_lib.php.inc | INOUT\_waiver\_signed.php.inc |
| INOUT\_lib.js | INOUT\_reports.js | PJKUTIL\_space\_cols.js |

In addition, the following file is used to “map” columns in the Excel cabin assignment spreadsheet to XML-compliant attribute names used in the Roster XML data file. It is found in the **~host/gniadmin/db** folder:

*INOUT\_map.xml*

The map file contains its own documentation and can be edited with any text or XML editor as needed. The map consists of <mapItem> elements, each corresponding to the “title” (first row) of a spreadsheet column. The element may contain a “req=’T’” attribute (to indicate that spreadsheet data must be present in this column), a “min=’99’” attribute to indicate the minimum length of data in a required field and a “zpadnb=’5’” field which indicates the data should be “leading-zero padded” if it is non-blank and less than the indicated length. The map need not list every column in the upload spreadsheet; those that aren’t listed are ignored. Also, the order of columns on the spreadsheet or in the map file is unimportant.

Each <mapItem> element contains 2 child elements, <xlCol> and <inout>. The <xlCol> elements indicate the actual “title” text used on the first row of the spreadsheet column. These can be changed as needed to reflect the actual entries on the spreadsheet. One of these should contain a “priKey=’T’” attribute to indicate which column contains the “primary key”, currently, the 3-character Confirmation Number. This field is used (rather than MemberKey) to enable “split” registrations where a member attends for a few days, leaves camp (i.e., does not pay for several days) then returns later for an additional stay (in this case, the InOut list will contain 2 entries for this member, each one documenting the arrival/departure of each of their stays).

The <inout> elements indicate the field names used by *INOUT\_main.php.inc* when constructing the attendance list. THEY SHOULD NOT BE CHANGED UNLESS CHANGED IN THE PHP MODULE. A “time=’T’” attribute is used to indicate any field (i.e., Arrive) that represents a date/time.

Each time a member is marked in or out, a transaction is written to **~host/gniadmin/db/INOUT\_txn\_db\_9999.xml** (where 9999 corresponds to the current year). This file is used to a) display the current In/Out status of an attendee and b) generate daily reports of activity/census.

When checking someone in, the operator should note if there is a check mark in the “Wavr” column for the member. If not, they should invoke the “Waiver Utilities” option on the popup In/Out window. This will query Smartwaiver to see if this member has a waiver on file and if so will mark the member accordingly. If there is no waiver on file for this member, the app will report this condition and the operator should select the “Present Waiver for Signing” option and have the member sign a waiver. When a waiver has been signed an entry containing this year’s confo code/Member Key/SmartWaiver waiverID is made in the *INOUT\_waiver\_db\_YYYY.xml* file.

**NOTE:** For new events (i.e., each year), *INOUT\_lib.php.inc* will, on startup, scan all old (prior year’s) *INOUT\_waiver\_db\_YYYY.xml* files to build a baseline of known waivers for all the participants in the new event. It processes these files oldest to newest so that the SmartWaiver ID of the most recently signed waiver is recorded for each member who has previously signed a waiver. If a registration comes in late, that is no problem, as the “names” db will be updated when the registration is processed, which will then be picked up the next time InOut is started.

If a member wishes to change their preference for, currently, photos they need only sign a new waiver with the correct settings. The most recent waiver on file takes precedence.

Also, when invoking the InOut tracker, you can specify a query string as follows: “**~host/inout?pjkdev**” to invoke an option which allows you to specify/pull up a previous year’s data.

## Smartwaiver

Smartwaiver itself provides secure, reliable storage of waivers, which are legal documents. There are some search/lookup utilities available on their site which can be used directly by logging into [www.smartwaiver.com](http://www.smartwaiver.com). There is also an option to download a PDF file of all waivers. They have a variety of pricing plans available. They currently offer a “Storage only” plan for $5/month that holds all signed waivers and allows for access through their API or website. However new waivers cannot be added. During the off-months, this is the plan being used. Just before a GNI event, the billing plan should be changed to (currently) “Monthly Business” which (currently) provides for up to 1,000 new waivers per month. At the conclusion of the event, the plan should be switched back to Storage Only. The plans are billed to a credit card.

The language and appearance of the waiver is managed within Smartwaiver’s website utilities. This is done by setting up a “waiver template” on the Smartwaiver site. The template currently in use is called “Naked Camp at the GNI Gathering”. Smartwaiver’s UI provides the tools to enter the text of the waiver and required and optional fields or selections. In addition, Smartwaiver provides an ability to “autofill” certain fields (specifically, first name, last name, date of birth and “tag”) when requesting a new waiver to be signed. The InOut app passes a member’s data to Smartwaiver when the “Present Waiver for Signing” option in the app is selected.

To access a Smartwaiver template, an API key is needed, along with the Template ID of the desired template. The API is unique to GNI and should never change. Unless a new template is created the template ID should never change either (the template can be edited without affecting the template ID). However, should you need to create a new waiver template, here is how to obtain the Template ID:

After creating a new waiver, go to the Waiver List page (Create/Edit Waivers in the SmartWaiver menu on the left), then click on the desired Waiver (don’t *Edit* it, just click on the waiver name). A new browser tab will open up with a link similar to this: https://waiver.smartwaiver.com/w/6111a94839f3d/web/. The Waiver ID is the number between /w/ and /web/, or, in this case, 6111a94839f3d.

Both of these values are managed in the *GNIREG Admin* module under the “*Configuration for Smartwaiver Interface”* option. In addition to obtaining the Waiver ID manually as described in the above paragraph, this configuration option can present a list of available waiver templates to permit selection and automatic coping of the Waiver Template ID. See *INOUT\_main.js* for more info. These values are generated within the Smartwaiver site.

Within a Smartwaiver, the “tag” field is a free-format text field can contain basically anything. We are using this as the linkage between GNI records and a member’s waiver through our unique Member Key, which is stored in this tag. Also stored in the tag is the current year’s “confo code” (3 letter random code) for this member. The aforementioned “Waiver Utilities” use this data to retrieve a particular member’s waiver. (Smartwaiver provides lookup by last name but, of course, that isn’t always unique. The process is to get a list of waivers for a given last name, then look through each one to match on the one containing the target Member Key.)

The only item of importance within the Smartwaiver console is the “Redirect URL”. It should contain:

[http://~HOST/inout/index.php?waiverid=[waiverid]&autotag=[autotag](http://~HOST/inout/index.php?waiverid=%5bwaiverid%5d&autotag=%5bautotag)]

*(where ~HOST is the URL of the current hosting site)*

This is the URL invoked after signing a waiver. In essence, once a waiver is signed, the InOut app is re-invoked with the waiver ID and tag information. When invoked, if the InOut app sees “waiverid” in the query string, that indicates that a waiver has been signed. This triggers an update to the InOut database with this information. In general, the only time it should be changed is for testing purposes. See also *INOUT\_waiver\_signed.php.inc.* The “Redirect URL” entry is (currently) found on the Smartwaiver site as follows:

* Find the option to “Create / Edit Waivers”.
* Select/edit the “Naked Camp at the GNI Gathering” template (currently the only one).
* Push the “Next” button at the bottom of the page (currently 3 times) until you get to the page which contains a sub-menu labeled “Redirect URL”.
* Open the “Redirect URL” sub-menu and set the “Website Version”/”Send participant to:” entry to the “production” URL as shown above.
* Also, under the “Emails” sub-menu, you can optionally specify that a copy of each signed waiver is automatically emailed to an address if desired.
* Continue to press “Next” at the bottom of the page until you get to “Publish Your Waiver”, then publish.

## Annual Maintenance Activities

* Switch Smartwaiver from a “Storage only” billing plan to “Monthly Business” (or whatever is appropriate)
* Ensure Smartwaiver redirect points to production (see above)
* Decide who, if anyone, should receive email notification of waiver signings (also see above; it had been set to send a copy of each signed waiver to PJK for test/verification purposes)
* Switch Smartwaiver into “Storage only” billing after the event
  + If desired, download a PDF file of the waivers generated for the event

# Paperless Registration/GNI-I4A Interface

The registration process automation has morphed over time and has undergone serious modification along the way. The code itself is fairly well documented. The original process was a bit ragged, partially due to the restrictions/limitations of our former DB management product, MemberClicks. The process has now been converted to a new vendor, Internet4Associations (I4A), which has a simpler, more power API.

In short, the process is as follows:

* Uploaded an Excel spreadsheet containing member registration data to our server
* Use the I4A API to update appropriate fields in each member’s I4A profile
* Send the member a confirmation email letting them know their registration is complete

The following modules, found in **~host/gniadmin**, comprise the registration process (see also the *GNIAdmin* section of this document):

|  |  |  |
| --- | --- | --- |
| CPCE\_check\_session.php.inc | GNIREG\_link.css | GNIREG\_link\_load\_cabin\_excel.php |
| CPCE\_classes.php.inc | GNIREG\_link\_cancellation.html.inc | GNIREG\_link\_load\_excel.html.inc |
| CPCE\_edit\_ids.html.inc | GNIREG\_link\_cancellation.js | GNIREG\_link\_load\_excel.js |
| CPCE\_edit\_ids.js | GNIREG\_link\_cancellation.php | GNIREG\_link\_load\_excel.php |
| CPCE\_edit\_ids.php | GNIREG\_link\_config.html.inc | GNIREG\_link\_load\_lib.php.inc |
| CPCE\_edit\_password.html.inc | GNIREG\_link\_config.js | GNIREG\_link\_main.html.inc |
| CPCE\_edit\_password.php | GNIREG\_link\_config.php | GNIREG\_link\_main.js |
| CPCE\_footer.html.inc | GNIREG\_link\_email.php.inc | GNIREG\_link\_main.php |
| CPCE\_globals.php.inc | GNIREG\_link\_email\_config.html.inc | GNIREG\_link\_user\_info.html.inc |
| CPCE\_header.html.inc | GNIREG\_link\_email\_config.js | GNIREG\_link\_user\_info.js |
| CPCE\_lib.php.inc | GNIREG\_link\_email\_config.php | GNIREG\_link\_user\_info.php |
| CPCE\_login.html.inc | GNIREG\_link\_foot.html.inc | GNIREG\_link\_waiver\_config.html.inc |
| CPCE\_style.css | GNIREG\_link\_head.html.inc | GNIREG\_link\_waiver\_config.php |
| CPCE\_system.php | GNIREG\_link\_I4A\_lib.php.inc | I4A custom GNI API.pdf |
| CPCE\_system\_ver.html.inc | GNIREG\_link\_inout\_utils.html.inc | index.php |
| CPCE\_system\_ver.php | GNIREG\_link\_inout\_utils.php | phpinfo.php |
| CPCE\_user\_lookup.php.inc | GNIREG\_link\_lib.js | PJKUTIL\_lib.js |
| GNIREG\_globals.php.inc | GNIREG\_link\_lib.php.inc |  |

There are also the following sub-folders in **~host/gniadmin**:

|  |  |
| --- | --- |
| Classes | This contains a library routine called *PHPExcel\_IOFactory.* This PHP utility provides access to the Excel data on the server side. It was originally downloaded from GitHub. |
| db | This contains all reference tables (i.e., XML configuration files) along with various “flat” database files for all the various sub-systems, currently:   |  |  | | --- | --- | | **CPCE\_users.php.inc** | This is file contains the user IDs/passwords (encrypted) for the GNIAdmin Utility menu. The file may be edited by hand if needed, provided the existing structure is maintained. Hint: to reset a password, replace the appropriate **$CPCE\_user\_pwd** entry with “PLAIN:newpwd” where “newpwd” is the new password. The system will automatically encrypt it the next time the user logs in. | | **GNI custom I4A API.pdf** | Document outlining the custom API created by I4A to enable GNI to update certain fields in the I4A database (their “public” API is read-only). | | **gni\_logo.jpg** | The GNI logo. This can reside anywhere. Most of these utilities reference the path to the logo in their **XXXXX\_globals.php.inc** file. The Registration Confirmation Email Configuration feature also contains the pathname which is used when generating confirmation emails. | | **GNIREG\_config\_list.xml** | This contains the currently defined “upload map” configuration sets. The file may be edited by hand providing, of course, that the XML syntax is maintained.  Each configuration set contains an id attribute, e.g., id=”1” (the number is unimportant; each one just needs to be unique). Each set is comprised of the following elements:   |  |  | | --- | --- | | **desc** | This is the name of the configuration set for information/identification purposes only (i.e., it is not used anywhere else) | | **ctrlMbrKey** | This contains the header on the spreadsheet column that contains Member Key | | **ctrlLName** | This contains the header on the spreadsheet column that contains the member’s last name (it is used when reporting during the upload process only) | | **mapItem** | There will be many of these. Each one will have an attribute of “type” which indicates the type of data being referenced (see *“Config For Registration/Cabin Assignment Upload”* above for more info on Type). Each **mapItem** element will contain 2 children:  **xlCol** – the header on the spreadsheet column that contains the target data to be uploaded  **i4aCol** – the I4A column name to receive the uploaded data | | | **GNIREG\_email\_config.xml** | The configuration that controls the generation of the paperless registration confirmation email (see that section for more information). | | **INOUT\_map.xml** | This maps data from the cabin assignment upload spreadsheet to the columns needed for the InOut Attendance Tracker. It can/must be edited by hand if needed (which will only happen if the spreadsheet column names are changed).  The map consists of multiple **mapItem** elements similar to those described for the **GNIREG\_config\_list.xml** above, with the following difference:   * There is no “type” attribute on the **mapItem** element * But there may be a “req” attribute which, when equal to ”T”, indicates that the given field is required (the upload program will complain if the required data is not found in the spreadsheet) | | **INOUT\_names\_db\_9999.xml** | This “DB” (actually an XML file) contains the names and other information about attendees for a given year’s event. This data is used by the InOut tracking app. It may be edited by hand, but any manual edits will be lost the next time any cabin assignments are uploaded.  There may be many of these files. “9999” refers to the year of a given event; each year’s event automatically generates a new file. Should multiple events be held the same year, simply rename or delete any existing files for that year. | | **INOUT\_txn\_db\_9999.xml** | Each time a member’s entry/exit is tracked by the InOut app an entry is appended to this file. As with the names\_db above, “9999” refers to the year of the event and the file may be deleted or renamed if there are multiple events in the same year.  Note that this is NOT an XML file, despite the filename. It is a “flat” transaction file where each line is comprised of 4 pipe-delimited (e.g., “|”) data element consisting of:  ConfoCode|Textual transaction time|UNIX timestamp transaction time|IN or OUT | | **INOUT\_waiver\_db\_9999.xml** | Similar to the txn\_db above, this file retains the SmartWaiver (see elsewhere) document ID for a member’s signed liability waiver. It consists of 3 pipe-delimited elements:  ConfoCode|MemberKey|SmartWaiver doc ID  Note that these will accumulate over the years and the InOut app will query all old waiver\_db files to see if a given member has a waiver on file. | | **MOBI\_config.xml** | This XML configuration file contains the following elements:   |  |  | | --- | --- | | **api\_key** | This is the API key for GNI’s access to the EventMobi schedule management app. This API is used to drive the Print Kiosk Schedule feature. Unless something changes, it never needs to be updated. | | **kiosk\_status** | Indicates whether the Print Kiosk is enabled or disabled | | **eventCode** | The EventMobi code for this year’s event (see the Print Kiosk section for more info) | | **event\_name** | The name of the event shown on the Print Kiosk display | | **upd\_freq** | The frequency, in seconds, that the Print Kiosk will query EventMobi for an updated schedule | | **map\_url** | The URL to an image (JPG, JPEG, PNG, etc.) to be presented as the campground map | | | **PJKUTIL\_lazyLoadSpin.gif** | An animated graphic of a spinning disk used during long load times | | **PVR Map.jpg** | An image to be shown in the Print Kiosk when requesting a map of the campground. This can reside anywhere. The **map\_url** described above points to this image. | | **ROSTER\_auth\_log.xml** | A temporary status file used when the Attendance Roster is active to prevent the session from dying. | | **ROSTER\_config.xml** | This XML configuration file controls the Attendance Roster generation and display. It contains the following elements:   |  |  | | --- | --- | | **eventName** | The title displayed on the Event Roster page | | **cabinCol** | The I4A column to be queried when collecting data for the generation of the Roster. The presence of data in the referenced field (e.g., c\_user\_gni\_gathering\_cabin\_no2021) will cause that member’s info to be selected and presented in the Roster.  NOTE: The member’s privacy settings in their I4A profile are respected. | | **autoUpd** | Indicates whether the Roster is periodically doing an update by querying the I4A database. | | **updFreq** | If autoUpd is ON, the frequency with which the Roster queries the I4A database, in minutes. | | | **ROSTER\_GNI.xml** | An XML file generated by the Attendance Roster that drives the Roster display. It is replaced each time an update occurs. | | **ROSTER\_usage\_9999.log** | Gathers statistics for the given Roster (e.g., “9999”) on the number of searches, prints, sorts, etc. For informational purposes. | |
| I4A sql | This contains the SQL source code used to generate the I4A “Views” that power the registration, roster and user lookup functions, currently:   |  |  | | --- | --- | | **pjk\_gni\_get\_member.sql** | Used to generate an I4A View of the same name to power the *“Lookup Member Info”* function on the GNIAdmin Utilities menu | | **pjk\_gni\_get\_roster.sql** | Used to generate and I4A View of the same name that extracts data for the Attendance Roster app on the Print Kiosk | | **pjk\_gni\_registraton\_baseline.sql** | Used to generate an I4A View of the same name that is invoked at the startup of the GNIAdmin Utilities to generate a table linking GNI Member Keys to I4A contactIDs; also retrieves a member’s I4A email and alternate email addresses. | |
| uploads | This directory is used to temporarily hold uploaded spreadsheet information. It is automatically created if necessary and automatically cleaned up when finished. |

## The Process

*GNIREG\_link\_load\_excel.php,* invoked from the *GNIAdmin* module, does the bulk of the work. The first step in this process is to update I4A user profiles to indicate:

* Attendance at the upcoming event (e.g., “2019 Gathering: 1”)
* The registration date
* Updates to other fields, such as the member’s “Nickname”
  + Nickname and other updates are a recent addition: if desired, additional fields can be updated from the spreadsheet by making the appropriate entry in the ***Current Config*** under the *“Config For Registration/Cabin Assignment Upload”* option.

The aforementioned spreadsheet is typically an extract from an Access database that contains pertinent information for registrants to this year’s event. Once one or a group of registrations have been processed, the spreadsheet is generated for those registrations and uploaded through the *GNIAdmin* module.

This process utilizes a custom API developed for GNI by I4A. A document outlining this utility (**GNI custom I4A API.pdf**) can be found in the **~host/gniadmin** folder.

While the custom API will accept either GNI’s Member Key or I4A’s contactID to identify the target record, unfortunately Member Key is not unique within GNI’s I4A data, and any members who appear more than once will cause the API to fail. Therefore, at the startup of the GNIAdmin utilities, a query is made to I4A for the Member Key, contactID, email and alternate email address of all active members. This information is then loaded into an associative array by Member Key which can then be used to provide the unique contactID when doing data upload with the I4A custom API.

## Annual Maintenance Activities

* Within the *GNIAdmin* module:
  + Use the *“Configuration for Registration Confirmation Email”* function to generate a sample confirmation email and update if needed/desired
  + Use *“Upload Registration Confirmation Info & Generate Paperless Confirmation Emails”* when processing new registrations

# Print Kiosk: Overview

The GNI Print Kiosk interfaces with GNI’s current event scheduling smart phone app, EventMobi ([www.eventmobi.com](http://www.eventmobi.com)), to enable printing of the daily schedule by attendees at GNI events. It also provides an option to print a map of the camp, as well as an Attendance Roster for the event.

The Kiosk itself is a SeePoint (<http://www.seepoint.com/>) CounterPoint countertop touch screen computer. It is currently running Windows 7 Professional. Internet access is through a USB wireless network adapter (the device has 4 USB ports). When in operation, the Kiosk typically will not have a keyboard/mouse attached. The physical ports are locked behind a security panel, but an external 4-port USB hub has been added so that the keyboard/mouse can be hooked up for maintenance without unscrewing the security panel.

The printer is (currently) an HP OfficeJet Pro 8715, but any fast printer (preferably able to do double sided printing) will work. The printer defaults (on the printer itself) should be set to Draft Printing and Double-Sided Printing to conserve ink and paper and speed printing. It is connected to the Kiosk via USB.

Firefox is installed on the Kiosk (the applications work best with this browser). The browser’s default home page should be the URL for the Print Kiosk/EventMobi app, currently: <https://www.clubpittsburgh.com/eventmobi>

There is also a Firefox-specific plug-in installed called “r\_kiosk”. This plug-in takes immediate control of the Kiosk upon boot, starts Firefox in full-screen mode and opens the Kiosk app. Unfortunately, this plug-in has since been discontinued (see <https://legacycollector.org/firefox-addons/1659/index.html>). However, it still does what we need it to do, so it is still in use. The only documentation available is here:

## **https://legacycollector.org/firefox-addons/1659/icon.pngR-kiosk**

**Author(s):**

* **Kimmo Heinaaro**

Real Kiosk is a Firefox 2.0 - 3.0 extension that defaults to full screen, disables all menus, toolbars, key commands and right button menus. Alt+Home still takes you home.   
  
You can enable Navigation toolbar by adding the following to user.js:  
user\_pref("rkiosk.navbar", true);  
  
You might want to remove the print dialog by adding following lines to your user.js:  
user\_pref("print.always\_print\_silent",true);  
user\_pref("print.show\_print\_progress",false);  
  
Notice that the user can still close Firefox with for example Alt-F4 and get access to your computer. You might want to prevent this with a suitable utility program for your operating system.   
  
Caution! R-kiosk extension can be removed only in Firefox Safe Mode. Howto: [http://kb.mozillazine.org/Safe\_Mode\_(Firefox)](http://kb.mozillazine.org/Safe_Mode_%28Firefox%29)

To close Firefox and access the OS, plug in the keyboard/mouse and press either Alt-F4 or the “Window” button (lower left corner). You can then close Firefox and access the system as usual.

# Print Kiosk: EventMobi

EventMobi provides an excellent API (documentation can be found on their website) to facilitate the print Kiosk. The Kiosk app works in real time. I.e., any changes made in the EventMobi app will show up after a browser refresh (the site automatically refreshes itself, currently every 10 minutes; the frequency can be adjusted in the Admin module). The Kiosk is actually pulling the event data from EventMobi on each refresh and formatting it for printing. It will print everything that is entered into EventMobi for the event description, including pictures.

The system has been tested it in all browsers except Opera and it works fine, with these caveats:

* The printed page header (with the date and column headers) repeats on every page in all browsers except Chrome & Safari. I’ve discovered that this is a known bug in Chrome; according to their website, someone started working on a fix in Feb of 2016… not sure about Safari. I could program around this but I assume we can select what browser we put on the kiosk.
* It will also work from an iPhone or iPad if it has an AirPort connected printer.

Code for the Print Kiosk resides at **~host/eventmobi.** The app is composed of the following software modules:

|  |  |  |
| --- | --- | --- |
| index.php | MOBI\_head.html.inc | MOBI\_main.js |
| MOBI\_admin.php | MOBI\_lazy\_load.php.inc | MOBI\_main.php.inc |
| MOBI\_foot.html.inc | MOBI\_lib.php.inc | MOBI\_style.css |
| MOBI\_globals.php.inc | MOBI\_main.html.inc | MOBI\_users.php.inc |

Each module is self-documented to describe its role/function. The basic logic flow is as follows: “index.php” determines if the Kiosk is “enabled” (controlled in the Admin module) and will put up a descriptive message if not (typically, the Kiosk is not enabled until the schedule has been at least partially entered into EventMobi).

If the Kiosk is active, the “lib” module will read a configuration file (MOBI\_config.xml, more on that below) to get EventMobi access information and will get a complete list of “sessions” for this year’s event as an XML document. A “session” in EventMobi’s parlance contains all information for a particular event: title, full description, start/end time, location, etc. (NOTE: the number of sessions is currently set to 750. If this number is exceeded, the Kiosk will complain. Comments in the “lib” module will direct a future developer how to handle the situation should it occur.)

The “main” server modules then analyze the XML document to dynamically determine the date range of the event. The “main.html” module will automatically format the correct number of tabs for the given number of event days. The session data is loaded into one large table with class names set on rows and columns to indicate the type of data, e.g., the date of the session, full description, and meal description. Then, when the user selects a particular date or type of data, the appropriate class display property is set to “none” or empty (“”) to hide/show the appropriate data. The server dynamically builds this table and generates the appropriate HTML.

However, at present, Internet performance in the Poconos is very bad at times. This has created problems at times whereby loading of all text and image data for all dates causes a long delay, which appears to the user as a frozen system, causing them to overload the system by making rapid, indiscriminate selections. To avoid this, a “lazy load” has been implemented such that:

1. Start/end times are encoded on the server to a total of 4 bytes each; JS on the browser does the decoding
2. Locations descriptions are only written the first time encountered; subsequent occurrences reference a numeric index identifier; JS decodes the index in the browser
3. While the entire table structure is generated by the server, only one day’s worth of data is written to it. When the user selects a different date, if that data hasn’t yet been loaded, an AJAX request is made to the server which then sends the requested data.

## Camp Map

If a URL pointing to an image (jpg, png, gif, bmp) of a map of camp has been specified in the configuration file, a button will appear at the top of the screen providing an ability to view/print the map. The image may be oriented as portrait or landscape; however, it will be displayed on the Kiosk however it is oriented. When printing, though, if the orientation is landscape, the server will generate a portrait version so that the printed version can expand to fill a full sheet (the “regular” schedule prints in portrait mode and it is not possible to programmatically instruct the printer to change orientation). The portrait version will be created on the server in the ~gni/eventmobi directory in a file called “Kiosk\_map\_portrait.jpg”. This file is generated/replaced (if the original is landscape) each time a print is requested.

## Attendance Roster Button

(See also *Print Kiosk: Attendance Roster* section elsewhere in this document)

If the Attendance Roster has been accessed previously during this session, an access button is presented at the top of the screen. If this is the first time the kiosk is being started up (or the session has expired; more on that below) the button will not appear. This was done so that a link to the Kiosk (i.e., the scheduled events) can be put on our public-facing website to give prospective attendees an idea of the activities available (we don’t want the general public to access the Roster, which contains private data for members only).

When the Kiosk is first activated for an event, the Roster button will not appear. To activate it, double touch (double click) the GNI logo at the top of the screen, then touch (click) the "Gay Naturists International, Inc." title. This will bring up the Roster app. You will be prompted to login (see *IDs/Passwords* section below). The system accepts any GNIAdmin Utility login credentials.

Sessions typically timeout after about 30 minutes, but since the system automatically refreshes every 10 minutes it will “remember” that the roster is available and present the button for the remainder of the event. (NOTE: if the system is down for longer than 30 minutes, it may be necessary to log in to the Roster again.)

Note to future developer: this “hidden” double/single touch to activate the button could be avoided by starting the Roster first…

## Admin Module/Configuration File

The schedule and map portion of the Kiosk is maintained from the *“EventMobi/Print Kiosk Configuration”* option on the GNIAdmin Main Menu. The administration options are as follows:

|  |  |
| --- | --- |
| Kiosk is currently: | Enabled or Disabled; if disabled, user will get an appropriate message |
| Event name: | The name to appear at the top of the Kiosk screen, e.g., “Naked Camp 2020”. This will need to be updated each year. |
| Event Code: | The EventMobi “code” for this event, e.g. 84b8f566-bf31-4b81-9d8f-fa4bbb51a5b3. So far, this code has remained constant from year to year, so it should not need to be changed. |
| API Key: | The EventMobi API key, e.g., 11D4PjPTfqU2iIHyXEzEe\_IpfP5A19gc8bEYG7S4kNx2zTENm5TUF9j3RJ5o8d2R. This should (most likely) never change; it is specific to our organization. |
| Update Freq (secs) : | How often, in seconds, the Kiosk will refresh (i.e., pull data from EventMobi) |
| Camp Map URL: | A fully-qualified URL (e.g., [http://www.clubpittsburgh.com/gniadmin/db/PVR Map.jpg](http://www.clubpittsburgh.com/gniadmin/db/PVR%20Map.jpg)) pointing to JPG, PNG, GIF or BMP image of a map of camp. This is not required and may be left blank. If blank, the Show/Print Map option will not be presented on the Kiosk. |

To change any of these parameters, simply press the “Edit” button. The parameters are stored in the “**~host/gniadmin/db**” directory in an XML file called **MOBI\_config.xml**. This is a flat file that can also be directly edited with any text or XML editor.

## Annual Maintenance Activities

As needed, use the *“EventMobi/Print Kiosk Configuration”* option to:

* Enable/disable access
* Update the “Event name”
* If changed, update EventMobi code/API key (not normal)
* If changed, update the Camp Map URL

After physically setting up the Print Kiosk (w/wireless dongle), printer, keyboard, mouse:

* Test the Print Kiosk
  + Check the Schedule information pulled from EventMobi
  + Don’t forget to "turn on" the Attendance Roster by double touching (double click) the GNI logo at the top of the page then touch (click) the "Gay Naturists International, Inc." title, then do initial login
* Ensure an adequate supply of ink/paper/staples
* Remove keyboard/mouse for production use

# Print Kiosk: Attendance Roster

The Attendance Roster provides a list of attendees at a given event with an ability to search, sort and print. It is typically “toggled to” via a button on the Schedule (EventMobi) Kiosk app. The Roster is driven by an XML database generated from an extract of I4A data. This is done in order to respect certain privacy settings that individual GNI members can make to their profile data (i.e., hide email address, hide entire profile, etc.). The Roster information is periodically updated based on the frequency set in the GNIAdmin module.

The Roster can be accessed directly at **~host/gniroster**. It is comprised of the following modules:

|  |  |  |
| --- | --- | --- |
| Index.php | ROSTER\_I4A\_upd.php | ROSTER\_main.js |
| PJKUTIL\_space\_cols.js | ROSTER\_lib.php.inc | ROSTER\_main.php.inc |
| ROSTER\_foot.html.inc | ROSTER\_login.html.inc | ROSTER\_style.css |
| ROSTER\_globals.php.inc | ROSTER\_login.php.inc |  |
| ROSTER\_head.html.inc | ROSTER\_main.html.inc |  |

Due to the sensitive nature of this data, the first time it is accessed an ID/Password challenge will be presented. When setting up the Print Kiosk, the operator will need to do an initial login. The IDs/Pwds used are the same as those used for the *GNIAdmin* module.

Upon login, a PHP session is started so the user can bounce between the Roster and Schedule without requiring a subsequent login. However, a PHP session will eventually time out due to inactivity (typically 30 minutes, but varies based on the local implementation), in which case, the ID/Pwd challenge would be presented again. When used on the Print Kiosk at a GNI event, this would present a problem: the application may go for hours of inactivity (e.g., overnight) and it would be unmanageable to continually have to check on the status of the login.

To address this, upon successful login, the **ROSTER\_login.php.inc** module writes a UUID (or AuthCode) to a cookie with a lifetime equal or greater than the duration of the entire event (currently hardcoded in ROSTER\_login.php.inc at 14 days). This UUID/lifespan is also written to **~host/gniadmin/db/ROSTER\_auth\_log.xml**. Upon subsequent login, the server (ROSTER\_login.php.inc) examines the client’s cookies for one that matches a known, unexpired AuthCode. If found the user is admitted; if not found or expired, the ID/Pwd challenge is presented.

There is also another complication: in typical operation, the XML database that drives the app is refreshed from I4A data based on the frequency set in the GNIAdmin module (currently 10 minutes). This periodic refresh is done:

1. To capture last minute attendees and/or changes to the attendee information (e.g., cabin assignments) and
2. To enable members to change their privacy settings (in I4A) and have those settings reflected in the Roster on a near-real-time basis.

NOTE: This periodic refresh is only done if there is a Roster page open (i.e., if the Kiosk is currently displaying the schedule, the Roster will not update until the next time it is visible).

However, it may be desirable to suspend this automatic update for a variety of reasons (e.g., problems with I4A). This can be accomplished in the GNIAdmin module. If auto update is disabled (or if an update is desired before the automatically scheduled one is due), the GNIAdmin module provides an option to “force” an immediate update.

## Roster Database Generation

The Roster data resides in an XML database at **~host/gniadmin/db/ROSTER\_GNI.xml**. This DB is overwritten each time the Roster auto-refreshes. It is important to note that until at least one cabin assignment is uploaded for an event, the Roster will present an error message indicating that no items could be found.

The process used to generate the XML database is to invoke the I4A API using the **pjk\_gni\_get\_roster** View. This process is automatically initiated by the Roster page based on the update frequency set in the GNIAdmin module.

While the update process is occurring, informative messages are displayed in the lower left corner of the window. When accessing the Roster for the first time for a given year, the list will be blank until an initial database build can be performed.

## Reporting

This module records the history of print usage in a file called **~host/gniadmin/db/ROSTER\_usage\_2019.log** (where the year corresponds to the current year). It is a simple flat file that records each time the roster is printed and what the print settings were used. It can be accessed/downloaded using FTP.

## Annual Maintenance Activities

Use the *GNIAdmin* module to:

* Update the ***Current Config*** under the *“Config For Registration/Cabin Assignment Upload”* option in the GNIAdmin module to reflect the current year/desired event name.
  + Note that even if a member is marked as attending this year’s event, if they don’t have a cabin assignment in I4A, they will be omitted from the Roster.
* Use the *“Turn ON/Off Auto Roster Update From I4A”* option as desired (should be on during the event).
* Change the update frequency if desired.

After physically setting up the Print Kiosk (w/wireless dongle), printer, keyboard, and mouse:

* Test the Roster
  + If needed, from the Event Schedule screen, don’t forget to log into the Roster by double touching (double click) the GNI logo at the top of the page then touch (click) the "Gay Naturists International, Inc." title, then do initial login.
* Ensure an adequate supply of ink/paper/staples.
* Remove keyboard/mouse for production use.

# Summary of Annual Maintenance/Update Procedures

## Paperless Registration Confirmation Annual Maintenance Activities

* Within the *GNIAdmin* module:
  + Use *“Config for Registration/Cabin Assignment Upload”* to update this year’s event/field names
  + Use *“Upload Registration Confirmation Info & Generate Confirmation Emails”* when processing new registrations

## Configuration for Registration Confirmation Email Annual Maintenance Activities

* Check/update the HTML email template document if needed
  + Generate a sample confirmation email and update if needed/desired
* Check/update the Bus Substitution Fields as needed

## Smartwaiver Annual Maintenance Activities

* Switch Smartwaiver from a “Storage only” billing plan to “Monthly Business” (or whatever is appropriate)
* Ensure Smartwaiver redirect points to production (see above)
* Decide who, if anyone, should receive email notification of waiver signings (also see above; it had been set to send a copy of each signed waiver to PJK for test/verification purposes)
* Switch Smartwaiver into “Storage only” billing after the event
  + If desired, download a PDF file of the waivers generated for the event

## Print Kiosk: Eventmobi/Schedule Annual Maintenance Activities

As needed, use the *“EventMobi/Print Kiosk Configuration”* option to:

* Enable/disable access
* Update the “Event name”
* Update the EventMobi Event Code to point to this year’s event
  + Unfortunately, EventMobi does NOT provide an ability to get this code online. (If you open the “Events List” on the EventMobi Admin page, it does show a column labeled “EVENT CODE”, but this is NOT the code that is used by the API.) To obtain the code for the current event, select “Integrations” from the EM Admin page (see IDs/Passwords at the end of this document), then select “Actions” for the GNI entry and then “Email Key”. This will generate an email containing the API Key and Event Codes.
* If changed, update the EventMobi API key (API update not normal)
* If changed, update the Camp Map URL

After physically setting up the Print Kiosk (w/wireless dongle), printer, keyboard, mouse:

* Test the Print Kiosk
  + Check the Schedule information pulled from EventMobi
  + Don’t forget to "turn on" the Attendance Roster by double touching (double click) the GNI logo at the top of the page then touch (click) the "Gay Naturists International, Inc." title, then do initial login
* Ensure an adequate supply of ink/paper/staples
* Remove keyboard/mouse for production use

## Print Kiosk: Attendance Roster Annual Maintenance Activities

Use the *GNIAdmin* module to:

* Update the ***Current Config*** under the “Config For Registration/Cabin Assignment Upload” option in the GNIAdmin module to reflect the current year/desired event name.
  + NOTE: This will most likely have already been done during the process of uploading this year’s registrations.
* *“Turn ON/Off Auto Roster Update From I4A”* as desired (should be on during the event).
* Change the update frequency if desired.

After physically setting up the Print Kiosk (w/wireless dongle), printer, keyboard, and mouse:

* Test the Roster
  + If needed, from the Event Schedule screen, don’t forget to log into the Roster by double touching (double click) the GNI logo at the top of the page then touch (click) the "Gay Naturists International, Inc." title, then do initial login.
* Ensure an adequate supply of ink/paper/staples.
* Remove keyboard/mouse for production use.

# IDs/Passwords

For ease of use (and also the ability to isolate/restrict this information) below is a collection of GNI’s current IDs/passwords for various 3rd party vendors and in-house subsystems:

FTP (to Source Code directories)

(www.clubpittsburgh.com, no encryption, port 21)

Username: gnipchost

Pwd: Gmainmen1!

I4A Admin

**https://www.gaynaturists.org**

pjkarlovich

152281822PJK

(or any admin ID/Pwd provided by I4A)

GNIAdmin Module

**~host/gniadmin**

peter

peter1

(or your own GNIAdmin ID/Pwd)

InOut Tracking Module

**~host/inout**

peter

peter1

Or any valid ID/Pwd for the *GNIAdmin* Module

Smartwaiver

**https://www.smartwaiver.com/**

gniadmin

GniGni1

EventMobi

**https://experience.eventmobi.com/**

Username: gni@gaynaturists.org

Pwd: Password00!

Print Kiosk Attendance Roster

**~host/gniroster**

peter

peter1

Or any valid ID/Pwd for the *GNIREG Admin* Module