ATHENARMS ADMINISTRATIONCONTENTS

INTRODUCTION

AthenaRMS is a request management system designed to provide a simple web interface for managing requests entered into a database via web forms or email. It is designed around the concept of a ticket object that includes the original request and default fields that define the problem type and ticket status.

AthenaRMS may be extended with additional fields ("attributes") to support the specific needs of your application. Instructions for customizing AthenaRMS are found in the section "CUSTOMIZING ATHENARMS USING TEMPLATES" below**.** This administrator's guide assumes you have installed the product already. It should provide you with enough information to set up new work queues ("instances") and customize them for your specific needs. For instructions on installation, see the file INSTALL in your docs directory of the distribution.

For additional questions not covered in the included documentation, try the AthenaRMS email lists, a list of which may be found at: http://athenarms.com/support.

CREATING NEW INSTANCES

OVERVIEW

Each work queue in AthenaRMS is called an "instance." Each instance may be customized with its own custom fields, support staff, email and web address, and domain name. Typically, a super admin will create a new instance from within the application, and then ask a system administrator to finish the installation by making the necessary changes to DNS, email settings, and web server settings.

Once the new instance has been created, the next steps are to customize the screens that make up the Athena application as necessary. AthenaRMS is written in PHP, and uses a templating model for customization. All Athena instances use a set of default templates that generate each of the application screens. Default templates can be overridden by copying them into the instance's custom template directory and making any necessary modifications. See the section of this document "CUSTOMIZING ATHENARMS USING TEMPLATES" for details.

To create a new AthenaRMS instance:

1. Log into an existing instance as a Super Admin. All admins will have access to an Admin menu option available in the top menu bar. From here, Super Admins can create new AthenaRMS instances, and add support people to an instance and modify their profiles.
2. Select the admin menu option, and then "Create or Edit an RMS." Scroll to the bottom of the page to create a new RMS instance. Note the checklist at the top of the "Create Another Request Management System" form."

If using a dedicated hostname:

a. Add the hostname to DNS, and point the IP at the Athena server

b. If using qmail:

  - Edit `/var/qmail/control/virtualdomains` and add an entry of the form: `domainname:Athena5`
  - Make sure you have a `/var/qmail/alias/.qmail-Athena5-default` file which contains: `|preline /path/to/php/bin/php -q /path/to/Athena5/lib/ handle_email.php`

Restart qmail-send if using the recommended installation directions. You can do this with `svc -t /service/qmail-send`

If using addresses on an existing email domain:

a. If using qmail, then for each RMS, make a file of the form `/var/qmail/alias/.qmail-RMS Short Name` which contains: `|preline /path/to/php/bin/php -q /path/to/Athena5/lib/ handle_email.php`. (After the first, you can use symlinks.)

Prepare the filesystem

`ln -s /path/to/Athena5/htdocs /path/into/your/webtree` (or make the necessary addition to your httpd.conf file)

Create the RMS by filling in the form with appropriate values for your RMS.

RMS SETTING DESCRIPTION

Customer: Not currently in use. Use "default"

RMS name: This is what you want to call your instance. This name will show up in the home page list of instances and in individual pages.

RMS Short name: Used in path name for custom templates.

Default Problem Name: An administrator can define a set of problem types for the RMS. This option allows the administrator to set the default(useful for requests sent via email).

Default Source Name: An administrator can define a set of sources for the instance. Sources track the origin of a request (email, web form1..n, call in, etc). This option allows the admin to set the default.

Default Email Source: Allows the admin to specify one of the above sources as the one representing requests submitted via email.

Email Domain: The administrator can assign the email domain used by the instance for receiving requests via email. See SETTING UP EMAIL.

Web Domain: Enter this as web.domain.com, not http://web.domain.com.

Web Path: Each RMS must have its own Web Domain/Web Path combination (must not be below another Athena path on a given domain).

Users URL: (`/users/browse`) You shouldn't have to change this

Confirmation URL: This is the URL of the confirmation page that is returned after submitting a form.

Email Signature: "Your friendly support staff", for instance.

Extended Email Signature: (can be empty, only shown below regular email signature on outbound emails)

New Ticket Notification: If set to "Y", then the default support person will be notified of new tickets.

Once the RMS has been created, add yourself as a support person by clicking on the name of the new RMS, then "Assign People to this RMS". You should also add additional support people, and define problem types and sources at this time.

INTEGRATING ATHENARMS WITH YOUR WEB SERVER

There are two basic ways to install Athena for the web: absorbing an entire hostname, or including under a specific path.

OPTION 1: WHOLE HOSTNAME

You run "example.com" and you want Athena to have a whole hostname all to itself (good for you!). Therefore you put into install.php the url "http://athena.example.com". You need to first make sure you actually create the DNS entry for "Athena.example.com" and that it points at your web server.

Next, configure your webserver so that `/` points at `/path/to/your/athena/install/htdocs`. For example, if you're running Apache and you installed Athena in `/usr/local/athena`, your httpd.conf for Apache might look like:

```
<VirtualHost *>

ServerName Athena.example.com

DocumentRoot /usr/local/Athena/htdocs

</VirtualHost>
```

OPTION 2: SPECIFIC PATH

You run "www.example.com" and host all sorts of company information on that site. You decide you want Athena stuff to be at http://www.example.com/support/, so that's what you told `install.php`. You have Athena installed in `/usr/local/Athena`, and your Apache config for www.example.com already has an entry like:

```
DocumentRoot /usr/local/www/data
```

So you do:

```
cd /usr/local/www/data

ln -s /usr/local/Athena/htdocs support
```

Now any access to http://www.example.com/support/ will be hitting Athena. You could also do this in the Apache config itself with: `Alias /support/ /usr/local/Athena/htdocs/`.

In either case, make sure for the Athena directory you have full override permissions and PHP turned on:

```
AllowOverride any
```

SETTING UP EMAIL

General instructions for integrating Athena with your email system are covered above, as well as in the installation procedures. Note that all mail for an email domain specified is parsed via the PHP script:

```
/path/to/athena5/lib/handle_email.php
```

You may want to read through this script for a full understanding of the process.

SUPPORT PERSONNEL

ADDING SUPPORT PERSONNEL

Only Super Admins have the privilege to add additional support personnel to an Athena instance and modify support privileges. As a Super Admin, select "Admin" to see a list of options for adding and modifying support personnel. The options are explained below:

1. Add New Support People: This option allows you to enter an entirely new support person record, including contact information, access rights, and whether or not the support person is added to either of the support drop-downs (explained below).

   NOTE: At this time you will have to create a new support person, and then edit the new account to set a user name and password. (This is a usability issue that will be fixed in an update to AthenaRMS.)

2. Edit Support People: This option allows you to modify all fields for an existing support person.
3. Add Existing People: Once a new person record has been created, super admins can add it to additional Athena instances by searching on first or last name.

MODIFYING SUPPORT PERSONNEL

When editing support people, there are five types of data that you can modify:

1. Contact information: name, email, address, phone, etc.
2. Login Access: There are three different types of access, based on the person type:
   - Support: Any person record that has login access to respond to tickets.
   - Support Admin: A support person that can edit default sources, and problem types, and add new attributes to an Athena instance.
   - Super Admin: A support admin that can also create and edit support accounts, and create new RMS instances.
3. In support drop-downs: Setting the "In Support Drop Down", or the "In Escalated Support Dropdown" flag to "Y", causes the support person to show up in the associated drop-downs on the "Edit Ticket" page within the Athena instance.
4. Ticket Access: There are currently three levels of access to a ticket as follows:
   - Y - Can edit all tickets.
   - N - Read only access to tickets.
   - R - Can edit only those tickets assigned to you.
5. User name and password: Allows the super admin to set a user name and password for the support account.

   NOTE: at this time user names must be alphanumeric. Numeric values are not allowed.

CUSTOMIZING ATHENARMS

ADDING EXTENDED ATTRIBUTES

AthenaRMS is designed to allow custom fields to be added to an Athena instance. AthenaRMS refers to these as "custom attributes.” The general process is to first add the custom attributes to the Athena instance via the admin screen, and then modify any custom templates that you want to refer to the attributes. Typically these include the list page, edit ticket page, and search page.

All attributes are added by support administrators by selecting "Admin/RMS Attributes" from the main navigation bar. When selected, the attribute screen will show a list of any existing extended attributes and their display types.

Adding attributes with predefined values

Attributes containing predefined values have display types of "single drop down,” "multi element drop down,” "radio button,” "checkbox.” To create these attributes, you first provide an attribute name (or string) in the RMS Attributes screen, then select the display type, and update the form. Next, you click on the associated ID for the new attribute, and provide a list of possible values.

For example, you may want to define an extra field called "priority,” with possible values of "low,” "medium,” and "high.” Furthermore, you probably want this field to show up as a pull-down menu option in the edit ticket screen.

To create this attribute, first add a new field named "priority" in the RMS Attribute screen and set the display type to "drop-down.” Next, click on the ID associated with the new "priority" attribute. In the following screen enter the "low,” "medium," and "high" attribute elements. Not that each attribute element that you add will be given its own ID as well. Make a mental note of these, as they will be used when modifying forms.

Adding free text attributes

Free text attributes are added exactly like those with predefined values. It is important to remember that even free text attributes require you to define an attribute element as a pointer to the value that will be entered as a free text entry.

For example, to create a text field called "My Dog,” first add the field name "My Dog" in the same way that you created a "priority" field above. This time select "text entry" as the display type. Update the data model by clicking on "Update.”

Next (don't forget this step), click on the ID associated with the new text field. This time, when adding attribute elements, simply type in some dummy name, like "text entry.” Then update the form. That's all there is to it. Accessing free-text entries is described below.

USING TEMPLATES

OVERVIEW

There are currently nine templates that can be customized on a per-Athena-instance basis. The PHP scripts below are referred to as the default templates and are located in the `/path/to/install/Athena5/templates` directory.

confirm.php This is the page that is displayed when a new ticket is submitted. This page will normally include a reference to the one-time URL provided for checking status on your ticket.

edit_ticket.php This is the screen that support staff accesses to review a ticket, add comments, and change status of a ticket. Usually this form will require customization to reflect any custom attributes added to the RMS instance.

edit_ticket_rnav.php This is an included file that is part of the edit ticket page.

input_form.php This is the default form used to submit tickets to the instance. Note that forms can be placed on any web site, but this form has the ability to access any of the instance-specific attributes, thus allowing for dynamic customization.

list.php This is the form used for displaying ticket status in list view. "my open," "all open," and search results all use this form .

search.php This is the search page for an AthenaRMS instance.

status.php This is the status page used to display ticket information based on a ticket's one-time URL.

status_update.php This is a popup form available from status.php that allows the ticket originator to add additional information about the ticket.

user_new_ticket_email_confirmation.php This is the email message that gets sent back to a user letting them know that a ticket was received and what the one-time URL for accessing ticket status is.

CUSTOMIZATION

Each template can be overridden with a local, instance-specific copy and customized to suit the needs of the Athena instance. Most default templates contain notes explaining what fields are readily available for use within the template. More sophisticated modifications can be made by tracing back through the core AthenaRMS libraries.

To make a custom template:

1. First, copy the form that you want to customize into your custom directory. A custom template directory is defined by the instance short name supplied when the instance was created. In our example, we will modify list.php

   ```
   cd /path/to/install/Athena5/templates/custom
   ```

   (this is the default location; you may have changed it in Athena.conf)

   ```
   mkdir SHORTNAME
   ```

   (SHORTNAME was entered when creating the new instance)

   ```
   cd SHORTNAME
   cp ../../list.php .
   ```

   (this will make a local copy of the "list.php" form. All other forms can be copied here as needed, and modified.)

2. Next, edit the local copy of the template.

ADDING STANDARD FIELDS AND ATTRIBUTES

In our example, list.php, we will add both a standard field and an extended attribute to demonstrate the difference.

The top of list.php explains the standard fields that are available to the script. You can add any of these values by including them directly. For instance, for each ticket in the list view listed, problem type would be referenced as:

```
$rec['problem_type']
```

If you wish to add the problem type to your list, simply add an extra column to the table header:

```
<td nowrap="nowrap" class="hdr-sm">&nbsp;Problem&nbsp;</td>
```

and then in the php section below, find the line:

```
foreach (array ('subject', 'first_name', 'last_name', 'status') as
$col) {
```

and add "`problem_type`", in the appropriate order.

ADDING EXTENDED ATTRIBUTES

Extended attributes are handled somewhat differently in Athena. They are implemented as a set of hash values referenced by their attribute IDs. If you are adding extended attributes to static forms that will be hosted outside of the AthenaRMS environment, simply view the page:

http://your.Athena.instance.com/admin/showall.php

View the source from this page, and copy verbatim into your form.

If you wish to add an extended attribute to one of the default forms, such as list.php, make sure that attribute_display.php is included in the template:

```
require_once 'attribute_display.php';
```

This library includes a set of methods for printing and editing extended attributes.

Next, instantiate new attribute display and value objects.

```
$ad = new Attribute_Display ($dbi); # get attribute names
$attv = new Attribute_Value ($dbi); # get attribute values
```

You can then add extended attributes to your form, by calling `members_by_id_att` with parameters for the ticket ID and the attribute ID as follows:

```
$ary = $attv->members_by_id_att ($rec[TICKET_ID], $attribute_id);
```

Where `$attribute_id` holds the value of any of the attributes listed in the RMS Attributes section of the admin area for Athena.

For instance, if you have an extended attribute that you call "priority,” with an associated attribute id = 10, then add the line: `$priority=10;` to the top of the form, and call `member_by_id_att` as:

```
$ary = $attv->members_by_id_att ($rec[TICKET_ID], $priority);
```

To print the value of the attribute, use: `$ary[0][MEMBER_NAME]` for attributes with predefined values (such as the priority attribute in this example) and `$ary[0][ADDITIONAL_INFO]` for extended attributes that are free text.

If you are adding a lot of extended attributes to a form, you may want to write a function to simplify the PHP code:

```
function print_extended_att($f,$ftype=0)
# ftype=0, then prepopulated, as in a drop down selection.
# ftype=1, then field is free text
{
global $attv,$rec;
$ary = $attv->members_by_id_att ($rec[TICKET_ID], $f);
switch ($ftype) {
case 0:
print "<td>".$ary[0][MEMBER_NAME]."</td>\n";
break;
case 1: print "<td>".$ary[0][ADDITIONAL_INFO]."</td>\n";
break;
default: print "<td> ?? </td>\n"; # a reminder in case you # forget something
}
}
```

You would then be able to add the extended attribute to your form by placing the following call into the appropriate place in your script: `print_extended_att($priority,"0");`

SPECIAL CONSIDERATIONS

SECURITY

There are a number of security considerations to make with any web application. We have noted a few here. You should perform your own required level of due diligence by searching the Internet for web security issues, including PHP, email, apache, and other topics.

registed_globals: AthenaRMS has been created and tested with the PHP configuration parameter "register_globals" turned off. This is the recommended way of configuring your PHP installation.

http authentication: Installation instructions have been provided for non-encrypted http connections. If you wish to configure with https, please make the necessary changes to your web server environment. See apache.org and openssl.org for details on setting up https.

Email: All email associated with AthenaRMS is sent clear text. If this is an issue, please consider adding email encryption options to the application and making them available to the AthenaRMS community.

THE SPECIAL INSTALLATION ADMIN ACCOUNT, $athena_admin

During installation, you were asked to set an email address for `$athena_admin`. This is a special account that gets messages for any type of email failure including:

1. New message or reply and the system can't figure out which RMS the message goes with:
    - Because the db is down
    - The message isn't addressed to any RMS email address
    - The system can't find the address in any of the headers (e.g. Blind copies, with qmail we check the delivered-to field as well)
2. Any other random delivery problems

BACKING UP YOUR DATA

At this time, you must rely on the native backup procedures for your DB installation. Back up often.

WHERE TO FIND OUT MORE

AthenaRMS 5.0 includes a help system that attempts to provide context for the entire application. We have built an Athena instance into the help system that allows users to submit questions, bug reports, and various feedback to the core Athena developers. While not all questions will get answered, the tool should prove to be a useful source of information for users and developers alike.

Media Net Link will also be providing Athena support contracts and managing them through the online help.

See http://athenarms.com/support for details.

EMAIL LISTS

A list of AthenaRMS email lists can be found at http://athenarms.com/support/.