RLM Activation Pro Manual

RLM v12.1

June, 2016

RLM Activation Pro Manual

V12.1

June, 2016

RLM Activation Pro™

Contents

RLM Activation Pro Manual

RLM Activation Pro Manual v12.0

RLM Activation Pro™

Welcome

This manual, the *RLM Activation Pro Manual*, is intended for organizations that have licensed the optional RLM Activation Pro software or the Hosted Activation Service from Reprise Software. This manual explains how to configure and run the activation software.

The RLM documentation is divided into 5 manuals:

Standard RLM Components

- *RLM Getting Started Guide* - an introduction to the basic concepts of license management and RLM
- *RLM Reference Manual* - the complete reference to all core RLM components
- *RLM License Administration Manual* - The License Administration manual, suitable for shipment to your customers

Optional RLM Components

- *RLM Activation Pro Getting Started Guide – An Introduction to the RLM Activation Pro software*
- *RLM Activation Pro Manual* **(this manual)**- Reference for the Optional RLM Activation Pro software

All five manuals are available at the Reprise Website:

http://www.reprisesoftware.com/kits/RLM_Getting_Started_Guide.pdf
http://www.reprisesoftware.com/kits/RLM_Reference.pdf
http://www.reprisesoftware.com/kits/RLM_License_Administration.pdf
http://www.reprisesoftware.com/kits/RLM_Activation_Pro_Getting_Started_Guide.pdf
http://www.reprisesoftware.com/kits/RLM_Activation_Pro.pdf

What's New in RLM Activation PRO v12.1

This section lists the new features in RLM Activation Pro v12.0 along with pointers to the relevant sections in the manual.

What's new

- ***Activation Pro now allows the merging of contacts and companies.*** See Customer List on page 37 for more information.

API changes

- The web services POST message to create an activation key will now create the actual key for you, using the standard Actpro algorithm, if you leave the "akey" parameter out in the request. See Appendix Q - Using Web Services to Access the ActPro Database on page 104 for more information.

- ***If rlm_activate() is called to request more licenses than remain on an activation key***, activation pro would formerly return RLM_ACT_KEY_USED. Beginning in RLM v12.1, activation pro now returns RLM_ACT_KEY_TOOMANY.

- ***The web services API now checks the user's access level*** (in addition to verifying that the account is a valid account). The user specified in the header must have *admin* or *edit* access in order to interact with the ActPro database. A *view* or *portal* access user will receive an insufficient privilege error.

Section 1 – RLM Activation Pro Basics

This section of the manual contains the information you need to set up and deploy the RLM Activation Pro software.

Activation Overview

Overview

RLM Activation Pro allows you to deliver a pre-generated **activation key** to your customer, and when they are ready to use your product, a transaction with the activation server running at your site allows the license to be fulfilled without manual intervention. When using activation, there is no need for you to get your user's hostid information - this is transmitted to the activation server automatically.

In the case of a node-locked product, a typical scenario would be that your customer runs the product on the desired machine, and if the license had not been fulfilled earlier, the product asks for an activation key. Once the activation key is supplied, the license is retrieved transparently. From this point on, the product runs with its license in place.

Floating licenses would operate in a similar manner, except that the number of floating licenses to be activated is required.

RLM Activation Pro utilizes a *MySQL* database for storing all activation data for quick access to large amounts of activation data. The rlc back office GUI is written in PHP and runs against the same databases as the activation license generator. The Activation Pro customer portal is also written in PHP and again, it runs against the same MySQL database.

The RLM kit contains a demo of the activation capability. For more information on this demo and how to run it, see the Activation Pro Getting Started Guide.

How Activation Works

RLM Activation is driven by two main data items - **Product Definitions** and **Activation Keys**. The RLM License Center tool (rlc) allows you to define various products to be activated.

Product definitions consist of a product definition name, a product name, version, expiration specification, and node-locked/floating indication. In addition, product definitions can contain additional attributes for the license. All product definition attributes are described in detail in the Product Definitions section of the Your Activation Database chapter on page 21.

Once a product is defined, you can add additional licenses to the product definition by editing that product definition. Once the product definition is complete, **activation keys** can be created. An **activation key** specifies the product to be fulfilled, the type of fulfillment, and how many fulfillments are allowed by this activation key. In addition, **activation keys** allow you to specify the license expiration and/or additional attributes for the license. If any of the attributes specified in the **activation key** are the same as attributes specified in the **product definition**, the **activation key** version of these attributes will be used. All activation key attributes are described in detail in the Your Activation Database chapter on page 21.

Once activation is set up, you give your customer an **activation key**, which they then use to retrieve the license for the particular machine which they are using. The license fulfillment happens via the internet, using the license generator (isvname_mklic) on your website (or on the Reprise hosted activation website if you are using our Hosted Activation product).

The request is made by your customer via either calls you make to the **activation request API** or the Reprise-supplied **activation GUI** built into the rlm webserver.

If you use the **activation request API** (rlm_activate()), you will get another opportunity to add additional attributes to the generated license. These attributes will override any specified in either the **activation key** or the **product definition**.

Note that the system date/time on the requesting computer must be within 7 days of the system clock on the activation server. If the clocks differ by more than 7 days, an RLM_ACT_CLIENT_TIME_BAD error status will be returned, and no license will be generated.

Activation Pro Components

RLM Activation Pro consists of 5 components:

- activation admin tool (rlc)
- activation server license generator (actpro_mklic + isvname.gen, or isvname_mklic)
- activation data
- activation request API (rlm_activate())
- optional activation GUI (built into the rlm webserver)

The activation admin tool, **RLM License Center** (rlc), allows you to set up the parameters for activation: **product definitions**, **activation keys**, **list of blacklisted hosts**, and other **setup data**, as well as view records of license fulfillments.

The activation server **license generator** (actpro_mklic + isvname.gen, or isvname_mklic) runs on your website (in the cgi-bin directory) and processes activation requests, recording the results in the activation database.

The **activation data** consists of product and activation key definitions, blacklisted host definitions, rlc setup data, as well as records of license fulfillments. The data is described in the **Activation Data** section below, and is all stored in a MySQL database.

The **activation request API** (rlm_activate() or the older rlm_act_request()) allows you to embed activation requests within your application or your installation tools.

The **optional activation GUI** is built into the RLM webserver (using the activation request API) to provide a generic activation user interface.

You will always be able to provide a simpler activation user interface to your customers by using the activation request API, since you know the answers to many of the questions which the Reprise-supplied activation GUI must ask. However, you can use the activation GUI in the rlm web interface to get started without modifying your applications.

Rehostable Licenses

Activation can be set up to support rehosting of licenses by the end user without ISV involvement. This involves the use of the rehostable hostid type, which is new in RLM v9.3. A rehostable hostid is requested at the time that rlm_activate() is called, and the activation server issues the license to this hostid. Later, if the user wishes to move this license to another system, the *rlm_act_revoke()* call can be used to revoke the rehostable license and allow it to be activated on another system. (Note that rehostable licenses work for nodelocked licenses only, and multiple-license product definitions cannot be activated with a rehostable hostid.).

The rehostable hostid is a copy-protected file, and if the system should crash, the hostid cannot be retrieved. As an ISV, you will need a policy to deal with this situation – e.g., to give the customer another activation in order to continue. Since this situation should be rare, a non-automated

request procedure might be quite sufficient. If you use the **Normal** fulfillment type with Rehostable licenses, the expiration date should be fixed. If you want an expiration date that is relative to the time of activation, you should use the **ReActivate** fulfillment type, and set the fulfillment count to the # of times you want to allow your customer to revoke and re-activate the license.

Why should **Normal** fulfillments use fixed license expirations? Well, when your customer revokes a rehostable license, the activation server returns the fulfillment count back to the activation key. If, for example, you issue 60-day licenses with **Normal** fulfillment type, your customer could fulfill the license, then on day 59 revoke that license and re-activate it and get a new 60-day license. If the activation key specifies a fixed expiration date, that is not an issue. Also, if you use a **ReActivate** fulfillment type, the original expiration date will be preserved on a new activation.

What happens if my customer doesn't have an internet connection?

For cases such as this, you can create a webpage so that your customer support people can activate the license for your customer, then email it to them. The RLM kit has an example html file that can be customized for this use. It is called ***activation_example.html*** and it is in the ***examples*** directory of the RLM kit. Modify this file for your use, and put it on an internal site for your support people to use.

Activation Use Cases

In the previous chapter we described what an activation system does and the architecture of RLM Activation Pro. This chapter will describe a number of use cases for activation.

The Simplest NodeLocked Activation Scenario

In it's simplest usage, you give an activation key to a customer and they use this key to retrieve their license.

In RLM ActPro ⇨

1. You create a product definition which defines the license to be created, specifying a nodelocked license.
2. Then, you create an activation key for this product. In most cases, you use a "normal" activation key.
3. When your customer purchases, you give them the activation key.
4. Inside your software, if you cannot check out a license, you put up a "please enter activation key" dialog box and prompt for the activation key.
5. Once you get the key from your user, call rlm_activate() to activate the license using the RLM ActPRO server. rlm_activate() returns the license as a text string.
6. Write the license to disk, then re-initialize RLM and check out the license.
7. Subsequent invocations of the product check out the license without going through the activation sequence.

Creating DEMO licenses automatically

If you are using nodelocked licenses, you can create demo licenses automatically quite easily with RLM Activation Pro. Assuming you are using the simple scenario above, proceed as follows.

In RLM ActPro ⇨

Create a demo product definition that specifies a limited duration, typically 30 or 60 days. Next, create an activation key with a fulfillment count of 0, which means unlimited fulfillments. This key would use the demo product definition you just created. Give the resulting activation key to all your prospects. When they activate with this key, they will get a license that runs for the number of days specified. If they attempt to re-activate with the same key on the same machine, the original license will be returned, ie, they will not be able to extend the demo period by re-activating on the same machine.

Disabling your software for a customer who did not pay

Sometimes, you create an activation key and deliver your software to a customer, but that customer does not pay. In this case, you can turn your software off using the following technique. We will assume you are using the simple scenario described earlier.

Once the customer has activated your software, and you decide that you no longer want them to be able to continue, take the following steps. This requires that you have saved the activation key somewhere that you can retrieve it, for example, in the product definition you could specify the "Include activation key in license?" checkbox.

In RLM ActPro ⇨

1. In RLC, disable the activation key in the RLM ActPro GUI (uncheck the "enabled" checkbox).
2. In your application, after the call to *rlm_checkout()* succeeds, call *rlm_license_akey()* to retrieve the activation key. Then call *rlm_act_info()* passing the activation key. If *rlm_act_info()* returns RLM_ACT_KEY_DISABLED, then the license is no longer valid. If *rlm_act_info()* returns 0, all is OK.

Subscription Licensing

If you sell your software using a subscription model, you might want to issue a long-term license and have the capability to disable the license at some point in the future if the customer cancels the subscription. You can use the following technique to accomplish this.

Once you issue a permanent license to a machine, it will work forever. However, you can use the activation server to disable the license if the customer no longer wants to pay for the subscription.

1. Your product definition can specify a long-term license (one year, two years, or as often as you are willing to give the customer a new activation key). In addition, the product definition should specify the "Include activation key in license?" checkbox.

In RLM ActPro ⇨

2. When your customer decides to cancel the subscription, in RLC, **disable** the activation key.
3. In your software, after the call to *rlm_checkout()* succeeds, call *rlm_license_akey()* and *rlm_license_hostid()* on the checked-out license.
4. Using the activation key and hostid from step (3), call *rlm_act_keyvalid()* passing the activation key and the hostid. If *rlm_act_keyvalid()* returns RLM_ACT_KEY_DISABLED, RLM_ACT_KEY_NOHOSTID, or RLM_ACT_KEY_HOSTID_REVOKED then the license is no longer valid. If *rlm_act_keyvalid()* returns 0, all is OK.

Dealing with decommissioned customer machines

Let's say that your customer activated a license then, sometime later, told you that the machine has been decommissioned. You allowed them to re-activate on a new machine. How do you insure they aren't using the original license?

In RLM ActPro ⇨

Once you issue a permanent license to a machine, it will work forever. However, you can use the activation server to detect that the customer continues to use the license on the old machine. This requires that you have saved the activation key somewhere that you can retrieve it.

1. When your customer tells you that the machine has been decommissioned, in RLC, **delete** the fulfillment for their activation key. This will allow them to activate using that activation key on a new machine.
2. In your software, after the call to *rlm_checkout()* succeeds, call *rlm_license_hostid()* on the checked-out license.
3. Using the hostid from step (2), call *rlm_act_keyvalid()* passing the activation key and the hostid. If *rlm_act_keyvalid()* returns RLM_ACT_KEY_DISABLED, RLM_ACT_KEY_NOHOSTID, or RLM_ACT_KEY_HOSTID_REVOKED then the license is no longer valid. If *rlm_act_keyvalid()* returns 0, all is OK.

Getting Started

First, you should read the Activation Overview section of this manual on page 7. Next, you should read the three setup chapters:

- Activation Pro Setup on page 15 (Note: if you are using Reprise Hosted Activation, you can skip this chapter – we have already set up Activation Pro for you).
- Your Activation Database on page 21 and
- Enabling Your Application for Activation on page 45.

When you have performed these steps, you are ready to issue licenses for your product with RLM Activation Pro.

To use the RLM Activation Pro administration tool (**rlc**), point your browser (RLM Activation Pro supports Firefox v3.6 or later, or Chrome v29.0 or later) to the URL where your system administrator set up the RLM Activation Pro software. The webserver will prompt you for your username and password. Once you provide these, you will see the initial **rlc** screen. Login to **rlc** using the username and password you received from your system administrator (the login area is in the top right-hand side of the screen).

Upgrading from an earlier version of Activation Pro

If you are currently running Activation Pro, you can easily upgrade to the new version. (Hosted Activation customers do not need to worry about software upgrades – we do these for you).

You should follow these steps:

1. Back up your MySQL database. On recent versions of Activation Pro, there is a backup command in the "Admin" tab, "Database" sub-tab. Select the backup file nam,e and press "Backup Database". If this command isn't in your version of activation pro or you prefer to backup outside Actpro, this can also be done with the phpMyAdmin program available as part of XAMPP and on most web hosting companies' admin pages. Select your database, click on the "export" tab, and press "GO" in the bottom-right hand corner. Save the file. While this step isn't strictly required, it is ***strongly recommended*** by Reprise Software.
2. Install the new Activation Pro software. Do this the same way you installed Activation Pro initially:
    - extract the kit.
    - Copy your isv generator settings file to the *actpro_setup* directory on the kit (see the next chapter for details).
    - Remove the actpro_setup directory on your webserver, if it is there (it shouldn't be).
    - ftp (or copy, if local) the "actpro_setup" directory and all it's contents to your website, as you did in the initial installation.
    - Browse to …../actpro_setup/setup.php on the webserver, just like when you first set up Activation Pro. The setup program will detect that you have an installation present (unless you renamed the "actpro" directory to something else. If you did, you can rename it back to "actpro" before you browse to setup.php), and will prompt you to upgrade. If you select "yes", it will then back up the program directory (actpro), install the new software into a new actpro directory, and upgrade the database from the old version to the new version.
    - Make sure that your activation pro license file is present in the cgi-bin directory on your webserver. Without this license file, when you attempt to activate, you will get the "RLM_EH_ACTPRO_UNLICENSED (-160)" error.

Activation Pro Setup

In order to set up Activation Pro, you will configure your webserver, then install the RLM Activation Pro software on the webserver. Once this is done, you can populate your database (as described in the Your Activation Database chapter on page 21), then add activation request calls to your application (described in the Enabling Your Application for Activation chapter on page 45).

If you use Reprise's hosted activation service, you can skip all of this chapter except the first part of step 3, below. You will create your generator settings file and upload it to the hosted activation server. The remainder of this chapter describes installation for ISVs who host their own Activation Pro server.

Requirements

Setting up *RLM Activation Pro* involves setting up a webserver with PHP and MySQL support, as well as the *RLM Activation Pro* software. Ideally, you should be familiar with downloading and installing open-source tools and using Web infrastructure.

RLM Activation Pro requires:

- The Apache Web server, version 2.2 or greater
- PHP version 5, with MySQL extensions
- MySQL server, version 5.1 or greater
- MySQL Connector/C development kit (you will only need this for linking your license generator and you only need to do this if you are using ISV-defined hostids)
- Mozilla Firefox v3.6 or later, or Chrome v29.0 or later (for running *rlc*, the administration tool)

To get you started as quickly and easily as possible with RLM Activation Pro we suggest that you install a pre-configured Apache/MySQL/PHP (AMP) stack. Reprise Software recommends the XAMPP stack from Apache Friends. This is the stack which we use to test Activation Pro. It is free, and it comes pre-built for all platforms on which we support RLM Activation Pro (although it is not appropriate for production, see below). While it is certainly possible for you to install Activation Pro on an existing Apache/MySQL/PHP stack, please understand that Reprise cannot support every combination of these components. We are familiar with the XAMPP stack, and will be able to offer help on XAMPP. We strongly urge you to install Activation Pro on an XAMPP stack to become familiar with the product, then, once you are familiar with RLM Activation Pro, move your installation to you ultimate web server or web hosting company. This way, you will have a working reference implementation. Note that XAMPP is not recommended for production installations, it is not secure. See here: http://www.apachefriends.org/en/xampp.html#300

If you are not comfortable setting up your own webserver, you might consider Reprise Software's hosted activation service, which will save you the trouble of installing even XAMPP.

Setting Up Activation Pro

To set up RLM Activation Pro, there are 7 steps:

1. Install a pre-configured Apache/MySQL/PHP (AMP) stack (described in this chapter).
2. Install the Activation Pro software on your webserver (described in this chapter).
3. Configure RLM Activation Pro (described in this chapter).
4. Install the Activation Pro license file you received from Reprise Software in the cgi-bin directory on your webserver.
5. (Optional) Configure your License Generator if you use an ISV-defined hostid.

6. Setup the Activation Database (described in the Your Activation Database chapter on page 21).
7. Add activation request calls to your application, or build a stand-alone activation utility (described in the Enabling Your Application for Activation chapter on page 45)

If you are using Activation Pro at a web hosting company, a different set of installation steps will be required. In particular, step 1 above isn't required. The differences will be outlined in the following sections.

If you are using Reprise Hosted Activation, we have done steps 1-3 for you, so unless you have an ISV-defined hostid (step 4), you can skip the remainder of this chapter and go straight to Your Activation Database on page 21.

Step 1. Install a pre-configured Apache/MySQL/PHP (AMP) stack

If you are installing *RLM Activation Pro* at your ISP, you can skip this section. If you are evaluating, however, Reprise Software recommends installing on a local AMP stack before you attempt to run *RLM Activation Pro* at your ISP.

To start, you need the stack of web services to support *RLM Activation Pro*. A pre-configured AMP stack called XAMPP is available for Windows and Linux systems at the Apache Friends website. To install it, do the following:

- For Windows:
    - Browse to: http://www.apachefriends.org/en/xampp-windows.html
    - Below the text "Jump-off point", select the XAMPP download.
    - select the "Installer" link. Save the file, then run it.
- For Linux:
    - Browse to http://www.apachefriends.org/en/xampp-linux.html
    - Follow the 4 steps on the download page
    - after installing, the package will be in /opt/lampp, execute the following commands (as root):
        - chmod 777 /opt/lampp/cgi-bin
        - chmod 777 /opt/lampp/htdocs

      (Note: these directories would not normally have 777 permissions, but we are doing this to make the rest of the setup easier. When you are finished, reset the permissions on cgi-bin and htdocs to their original settings).
        - Now you can log out as root.

Step 2. Install the *RLM Activation Pro* kit from the Reprise website

RLM Activation Pro operates on a particular RLM binary platform, and is supported on windows, and linux systems (x86_w and x86_l):

On all Platforms

First, download the Activation Pro kit: go to the Reprise Website Download area, enter your username and password, and select the "RLM Activation Pro" kit. Pick the appropriate platform, and save this on your system.

Note: When downloading Unix kits using Internet Explorer on Windows XP systems, the files are incorrectly named as 'actpro.platform.tar.tar', rather than 'actpro.platform.tar.gz', once downloaded. This is a browser issue - after transfer, please rename the file before installation. Or, better yet, use Firefox or Chrome.

Also note: if you are installing on your ISP's webserver, be sure to chose the *RLM Activation Pro* kit appropriate for your ISP's operating system.

Continuing the Installation on Linux

Extract the kit files:

Use gunzip/tar to extract the archive:

```
% gunzip actpro.platform.tar.gz
% tar xvf actpro.platform.tar
```

Continuing the Installation on Windows

Extract the kit files:

On Windows, the kit is in a ZIP file. Unzip the file in the desired location.

Step 3. Configure *RLM Activation Pro*

- **Copy your License Generator Settings file from your RLM kit.**

  You can create the license generator settings file (*isvname*.gen) on any supported RLM platform. To create it, execute the following command in the configured RLM kit area:

  rlmsign -generator

  Copy the resulting file(*isvname*.gen) to the main actpro directory (e.g. actpro.x86_l2) **and** the *actpro_setup* directory. If you are copying between Unix and Windows systems, be sure that your copy program doesn't insert carriage-control characters. The easiest way to make sure of this is to check the file size to make sure it is the same after copying.

- **Install the *RLM Activation Pro* files on your webserver**

  - FTP the contents of the actpro_setup directory to your ISP's webserver space, e.g. *//www.yourdomain.com/actpro_setup (or copy to your webserver if it is local)*
  - Make sure that the *actpro_setup* directory is mode rwx to the web user.
  - Make sure that *actpro_setup/rlmact.mysql* is writable by the web user.

- **Run the Activation Pro setup program.**
    - On Windows: Run the XAMPP control panel. Start the Apache and MySQL services. On Linux, this happened as part of installation. If you are running at your ISP, you can skip this step.
    - Point your browser (Firefox or Chrome recommended) to:

      http://localhost/actpro_setup/setup.php

    - The defaults in the form should be correct, so press "Setup activation database". ***REMEMBER the admin account username and password – you will need this later***. (The default is admin and admin).

NOTE: if you are going to host your servers at your ISP, many ISPs have naming conventions for MySQL databases, and, in addition, many ISPs will not allow you to create a new database from a script, the way that *Activation Pro's* setup.php software does it. If this is the case, simply create the database manually using your ISP's administration tools, then run setup.php after the database is created.

Security Considerations

Once your Activation Pro software is installed, you will want to check the permissions on all files and perform other security checks on your website. While this list is not complete, it is a good start:

1. Remove the directory *actpro_setup* on your webserver. You no longer need this. (Note that setup normally does this).
2. Change the modes of the following files, as shown (note: the setup program attempts to do this, but on some servers the file modes are not set correctly):

   all .php files – mode 0400

   rlmact.mysql – mode 400

   all other files (including all files in the images directory) – mode 0444
3. Make sure that your cgi-bin/isvname_mklic program (or isvname.gen file) is read-execute to owner, ie, mode 500.
4. You should set up a .htaccess file to protect access to the *actpro* directory. Many ISPs have a tool to password-protect the directory. If they don't you can get help here: http://www.htaccessredirect.net/
5. Reset permissions of the cgi-bin and htdocs directories to their original settings.

Step 4. *Install the Activation Pro license you received from Reprise Software*

- You will receive an email from Reprise Software with your Activation Pro license. Create a file named *something*.lic (typically *isvname*.lic) containing the LICENSE line(s) from this email. We will use this file in the next step.
- Once you have installed the Activation Pro software, then log in as an *admin* user in the GUI, and select the "Admin" tab. In the 2nd row of tabs, select "ActPro License" . On

this screen, you can browse to the license file you created , and upload the file to the webserver using the “Upload File” button.

You will need to update this license annually when you receive your new Activation Pro license from Reprise Software.

Step 5 (OPTIONAL) - Configuring your license generator if you use an ISV-defined hostid

If your software uses an RLM ISV-defined hostid, you will need to build a custom license generator for *Activation Pro*. In order to do this, you need the MySQL Connector/C development environment.

To install the Connector/C development environment:

```
1. Browse to: http://dev.mysql.com/downloads/connector/c/

2. Select the appropriate platform from the drop-down list

3. Select the most appropriate distribution for your operating system,
and download that version.  Follow the directions for installation.
Note that you will have to register to download the MySQL software).
Select a mirror site and download the binaries.  Save the file.

4. Install the connector kit in a convenient location.
```

For example, on windows, you might download the Windows (x86, 32-bit), MSI Installer. Select the mirror site, then download. Once downloaded, start the installer by double-clicking on it's icon. Click past the security warning, accept the license agreement, then click “next”, and select a “custom” installation, unless you want to put the kit in the standard location. Click “Next”, then “Install”.

On Linux, you might pick the “Linux-Generic” platform, and the Linux ver. .3.glibc2 (x86, 32-bit), Compressed TAR Archive. Select the mirror site, then download, save the file. Once downloaded, gunzip the installer, and change directory to where you want to install the MySQL software. Extract the installer with tar. Click past the security warning, accept the license agreement, then click “next”, and select a “custom” installation (Unless you want to use the default location). Click “OK”. Click “Next”, then “Install".

Once the Connector/C development kit is installed, do the following:

```
1. edit the makefile in the Activation Pro kit and change the
definition of MYSQLDEV to be the directory where you installed the
Connector/C kit.

2, edit the makefile and change the definition of ISV_DEF_HOSTID_OBJS
to point to your object files that implement your ISV-defined hostid.

3.Copy the RLM library (rlm.a or rlmclient.lib on windows) from a
production RLM kit of the correct platform.

4. create the custom generator by typing "make isvname_mklic" where
isvname is your ISV name.

5. Copy the isvname_mklic binary to your webserver cgi-bin directory,
replacing the generic generator which was installed earlier when the
kit was installed, or, if you have not installed the files yet, copy
isvname_mklic to the actpro_setup directory before ftping the
actpro_setup directory to your webserver.  (Note: Additionally, the
rlc admin account has an Administer Database menu item which contains
```

```
an option to upload the license generator from your local system to
the webserver cgi-bin directory.)
```

Using rlc with https

rlc can be configured to use the https:// protocol. To do this, copy the file ***htaccess.https*** in the actpro directory on your webserver to ***.htaccess*** in the same directory. You may get a certificate warning message the first time you connect to the site, which you can ignore. If you want to fix the certificate warning, you will need to purchase an SSL certificate for your website.

Note that the license generator itself, ISV_mklic, will not work with https. The traffic from the rlm_act_request() call to the license generator is already encrypted (as of RLM v9.0), so activation keys are not sent in the clear.

Your Activation Database

Once you have installed and configured your Activation Pro software (as described in the Activation Pro Setup chapter on page 15), you are ready to set up your Activation Database.

All RLM activation activities are controlled by a database of **product definitions** and **activation key definitions**. In addition, you can create a **blacklist** of domains which are not allowed to activate licenses.

Before you can begin to use RLM activation, you must set up at least one product definition and one activation key definition, as described below.

These operations are performed by **rlc** - RLM License Center. **rlc** is a web-based application which allows you to maintain your activation database.

rlc uses user access permissions to control access to the three main functions:

- viewing and reporting on the activation database (**View** Access)
- editing the activation database (**Edit** Access)
- administering users/permissions/database (**Admin** Access)

Running rlc

You run rlc by pointing your browser to the URL where you installed the Activation Pro software. The server uses a MySQL database to store all the activation data.

rlc Access Control

Access control is performed by logging in as the **admin** user and creating individual user accounts. An account has one of 4 levels of access:

- View
- Edit
- Admin
- Portal (similar to View, for your customers)

An account with **View** access can view the activation data, but cannot create or edit any data.

An account with **Edit** access can do anything an account with **View** access can do, as well as edit any data except user accounts.

An account with **Admin** access can do anything an account with **Edit** access can do, as well as create and delete users, upload the activation license generator settings and perform other administration functions.

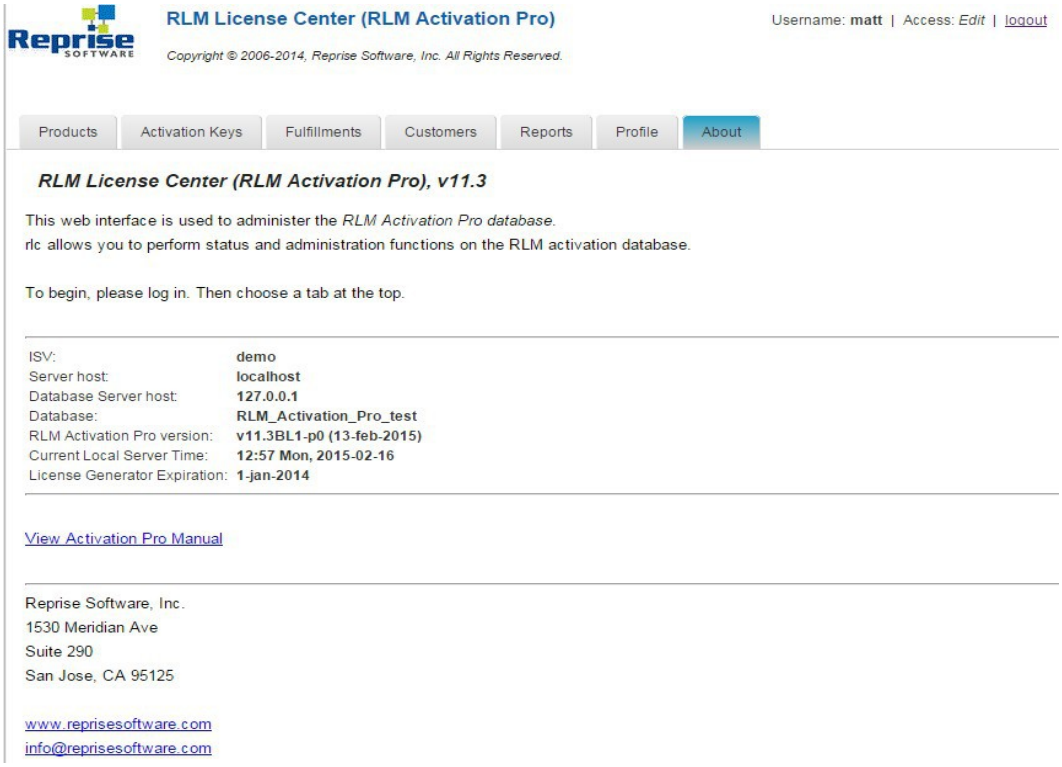
An account with **Portal** access can view activation keys and fulfillments associated with the contact's company – via the separate portal software (see Customer Portal on page 51). In addition, users with Portal access can only log in to the customer portal, not to the main Actpro administrative site (this was changed in Activation Pro v11.3).

Using rlc

There are 3 main activities you will perform with **rlc**:

- initial setup of your activation database
- viewing and/or updating the activation data
- generating individual licenses

When you run **rlc**, your browser will display a page that has a title and login area at the top, along with a row of tabs just below, and a main display area below the tabs, as shown below:

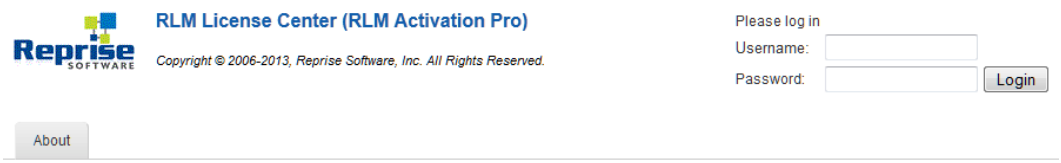


The tabs are arranged by activation data and functions, as described:

- Products – view or edit product definitions, and create new ones.
- Activation Keys – view/edit activation keys and create new ones.
- Fulfillments – view fulfillments, and delete fulfillment data.
- Customers – list contacts/companies, edit, and add new ones.
- Reports – Generate reports on activation activity.
- Admin (admin users only) – administer the Activation Pro system
- Profile – edit your account data.
- About – lists information about Actpro (as shown above)

Logging In

If you are not logged in, at the top right-hand side of the screen, username and password boxes are displayed, along with a **Login** button. Enter the username and password assigned by your administrator (when Activation Pro is installed, an admin user is created with password "admin".):



Once logged in, this area shows your username, access level, and a logout link.

Please note that you can only log in to one account from a single browser at the same time, and that it is also not possible to log into the same account from different browsers at the same time – if you log into the same account again, the first login will be effectively logged out.

To change your password, select the "Profile" tab, then press **"Change Password".** (Note: passwords can consist of numbers, lowercase letters and uppercase letters only).

Logout logs you out.

You will want to have non-administrator users for the day-to-day operations of managing your activation server. To create these user(s) log in to the activation server as the *admin* user, then create a second user who can make changes to the activation database. Perform the following steps:

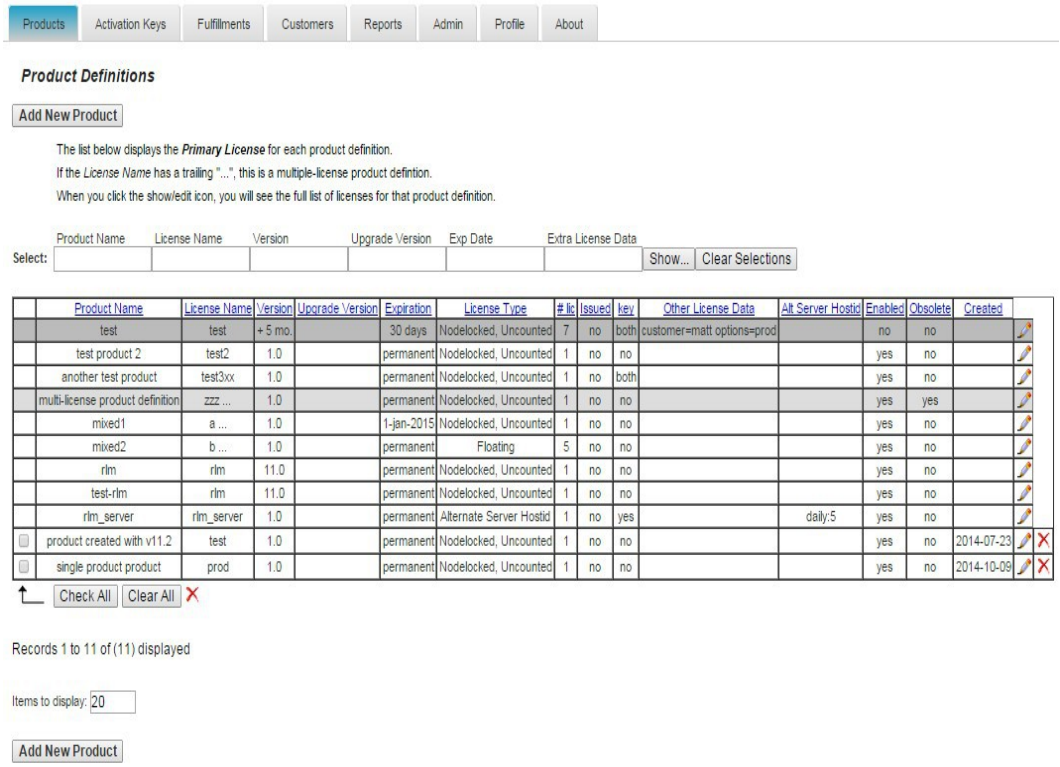
1. Point your browser to your installed version of Activation Pro, for example: http://localhost/actpro/index.php
2. On the right-hand side of the screen, near the top, type "admin" in the Username box, and "admin" in the password box. Press "login". (This assumes you did not change the default admin user's password).
3. Select the "Admin" tab, and the "Users" tab on the 2nd row.
4. Press the "**Create New User**" button below the list of users.
5. Select a username, Contact (if appropriate), email, and password, and "Edit" access in the "Create New User" page, then press the "**Create User**" button. Go back to the admin page by clicking on the "Admin Page" link. You should now see your new user, with Edit access.
6. Log out (at the top right).

Minimum Setup

The minimal setup required to use Activation Pro is one product definition and one activation key. The next 2 sections describe how you create a product definition and activation key.

Product Definitions

Viewing, Editing, and Creating new products is done in the "Products" tab, as shown here. When you press the "Products" tab, you will see a list of product definitions – these are all the product definitions in your Activation Pro system (more on this list later):

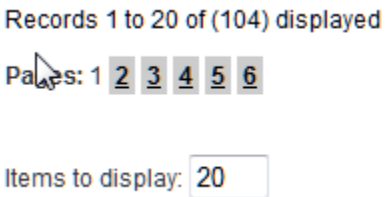


(Note that only "**product created with v11.2**" and **"single product product"** have a creation date. All products created prior to v11.2 will have blank creation dates).

At the bottom of the list there are 2 buttons ("Check All" and "Clear All"), and a red X:

Every product definition which can be deleted (ie, product definitions with the red X on the right-hand side of the list) will have a checkbox on the left. If you would like to delete multiple product definitions in a single operation, check the boxes corresponding to these products then click the red X at the bottom of the form. Note that you cannot delete a product definition which has any activation keys defined using it, or any product definition that has multiple licenses associated. To delete a product definition with multiple licenses, edit the product and delete all the non-primary licenses first, then you can delete it in the main product definition screen.

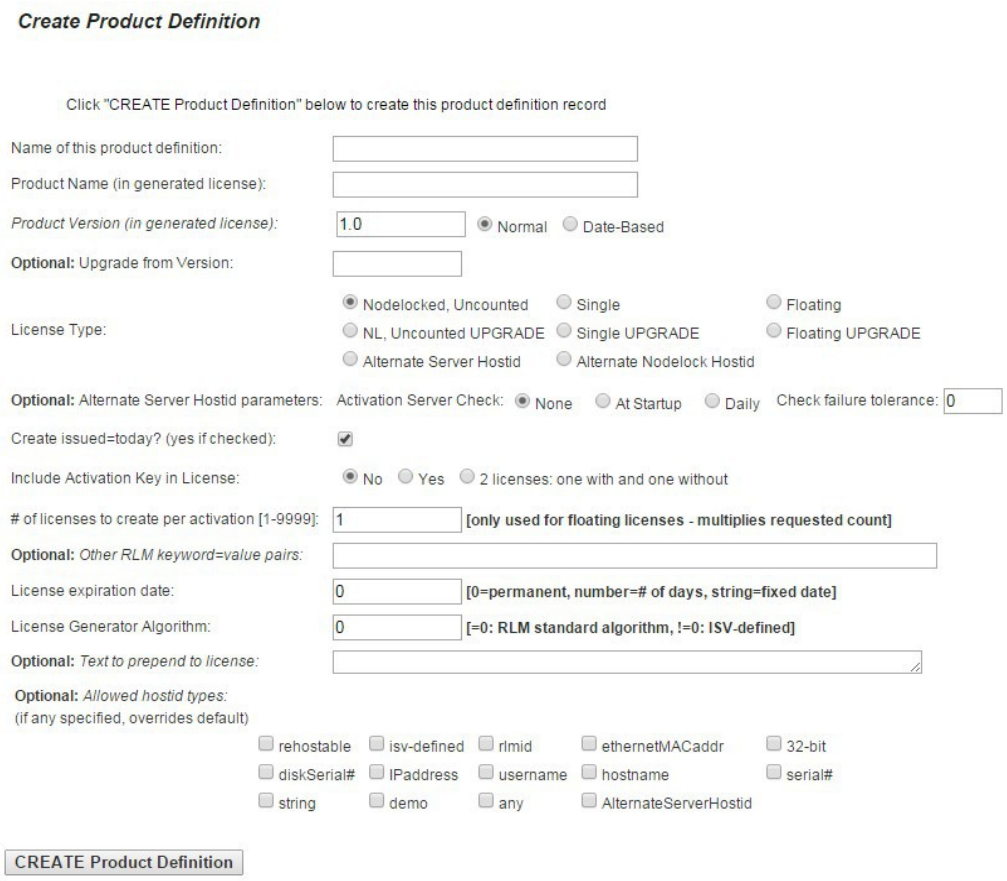
You can control the number of product definitions displayed by changing the "Items to display" count at the bottom of the screen. You can also filter the list by entering text in the "Select:" text boxes at the top of the screen. If the list is longer than the number of items to display, RLC will display a pagination box at the end of the list like the following:



The list filtering and pagination are common to all the tabs: **Products**, **Activation Keys**, **Fulfillments, Customers,** and **Reports**.

Product Definitions which are disabled (not deleted) will appear in gray in the list, and can be re-enabled later.

Everything in rlc is driven from *product definitions*, so you should create at least one product definition first. Product definition allows you to specify a name for the "product" which is then associated with a license product name, version, and several other license attributes. When you press the **Add New Product** button, you will see the screen below:



A product definitions consist of:

- a product definition name
- a product (license) name, i.e. what you check out in your application.
- a product version
- an optional upgrade-from version
- a license type
- optional Alternate Server Hostid parameters
- an issued= flag
- an akey= specification
- a license multiplier (for floating licenses only)
- optional additional attributes for the license
- a product expiration specification
- the license generator algorithm
- optional text to prepend to the license (for primary products only)
- a list of allowed hostid types, if you want to override the default

The **product definition name** is the name used to refer to this product definition. It is not the name of the product license generated by RLM activation. This might be something like "Evaluation Office License", in which case the actual product name might be "Office".

The **product name** is the name of the RLM license to be generated. This is the name that appears on the LICENSE line in your RLM license file.

The **product version** is the version of the product on the generated LICENSE line. If you want to use a fixed product version, click the "Normal" radio button and enter the version number in the text box to the right of that button. This version must be a valid RLM version number, i.e. it must be a number in the format of a floating-point number, of less than 10 total characters. An example valid RLM version is 1.0 or 1000.27

If you want to use a date-based version, click the "Date-Based" radio button, and enter the number of months after activation in the text box. A date-based version is of the form YYYY.MM (See "Maintenance-Thru-Date License" in the RLM Reference manual chapter on "License Models" for a description of how date-based versions work). So, for example, if you select 9 in the text box and the license is activated in January 2014, the resulting version will be "2014.10".

The **upgrade from version** is the minimum version of the old product which is eligible for upgrades in an RLM UPGRADE license.

The **license type** controls the type of generated license. Choices are:

- node-locked, uncounted
- single
- floating
- upgrade of any of the 3 types above
- Alternate Server Hostid (new in RLM v11.2)

In the case of floating licenses, the requested count is the number of floating licenses generated. The requested count must be less than or equal to the remaining fulfillment count available in the activation key. (As a special case, a request for 0 licenses will fulfill all remaining licenses from the activation key). For the other license types, the requested count is ignored.

For Alternate Server Hostid licenses, you must also fill in the optional "Alternate Server Hostid" parameters – the serial #, Check interval, and type. When an Alternate Server Hostid license is generated, both the HOST/ISV lines and the license line will be generated. Note that Alternate Server Hostid licenses utilize the license "options" and "issuer" fields, so you should not specify either of these in the "Other RLM keyword=value pairs" field. Also, you will most likely want to set "Include Activation Key in License" to "yes" when using this license type. Also, if you are using Alternate Server Hostids to provide rehostable licenses for your license server, you will want to select only the "rehostable" license type in the "Allowed hostid types" selection boxes at the bottom of the form, either in the product definition or in the activation key.

Finally, when you create an alternate server hostid key, you almost certainly want to give it an activation count of 1. If you give it a count > 1, your user can create multiple servers which have the *same hostid.* This means that any license you issue to this hostid will work on each of the servers so activated.

For more details on Alternate Server Hostids, please see the chapter titled "Alternate Server Hostids" in the RLM Reference Manual.

The **Alternate Server Hostid parameters** are used to specify the parameters required for an Alternate Server Hostid-type license. They are unused for all other license types. See the RLM Reference Manual for the meaning of these parameters.

The **Create issued=today?** checkbox controls whether the generated license contains an *issued=today* date attribute.

The **Include activation key in license** radio buttons control whether the generated license contains an *akey=activation_key* attribute. The akey= attribute was added in RLM v11.0. If you deliver software built with pre-v11, the software won't recognize the akey= license attribute, and the license will be invalid. By selecting 2 licenses (3rd choice), ActPro will generate one license with the akey= attribute and a 2nd license without the akey= attribute which will be usable by your pre-v11 applications. Note that a second license will NEVER be generated for any counted license type, since this would yield extra licenses. **Also note that RLM has a 35-character limit for activation keys in the license, so while activation pro will allow an activation key up to 60 characters, you will need to limit the length of your activation keys to 35 characters if you intend to use them in the *akey*= license attribute in rlm.**

The **Number of Licenses to Create per Activation** is a multiplier used for floating licenses only. This number is multiplied by the requested count in the activation request to yield the total number of licenses created for this product name. For example, if the activation request is for a count of 7, and the number of licenses to create per activation is 5, the license count for this product name will be 35. You should note that the requested count is the amount deducted from the activation key's remaining count field, NOT the resulting license count from this computation. In other words, for this example, 7 would be deducted from the activation key's remaining count, not 35.

The **other license parameters** are additional RLM license attributes which will be placed into the generated LICENSE. These must be valid RLM license keyword/value pairs, e.g. "options=professional". Any options specified in the activation key override corresponding attributes in the product definition, thereby allowing you to have default attributes which you modify for a particular customer when you create their activation key. Also, if you specify other license parameters in your call to *rlm_activate()* (or *rlm_act_request()*) those parameters will override both the product definition and the activation key.

The **License expiration date** controls the expiration date for the generated LICENSE line. This is one of:

- 0 (or blank or "permanant") - for a permanent license
- an integer - the number of days from activation to expiration (ie, 30 means the license expires 30 days after it is activated)
- an actual expiration date, in the form 1-jan-2013 - the actual expiration date, independent of the date of activation, in the standard RLM expiration date format. Beginning in Actpro v12.0, you can specify a fixed expiration date as YYYY-MM-DD, for example 2016-05-23 for May 23, 2016. The actpro license generator will convert this date to the standard RLM format when the license is created. You can convert all expiration dates in your database to this format using the "Normalize Dates" function in the Admin/Database tab.

If you have an alternate license generator, specify the index of this generator in the **License Generator Algorithm**. This value should normally be left set to 0 – the default Activation Pro license generator.

You can cause the license generator to prepend up to 1024 bytes of text to the license generated (for primary products only). This text can be any fixed string you want (note restriction below), but you should put a pound sign ('#') at the beginning of each line so that you don't confuse the RLM license parser. Place your text in the **Text to prepend to license** text box. Also note that your text ***cannot contain the "LICENSE" keyword*** (you can use the word "license" as long as it is not in all uppercase)**.**

If you want to override the list of allowed hostids specified in rlm_isv_config.c with the *rlm_isv_cfg_actpro_allowed_hostids()* call, you can select a set of allowed hostids in the **Allowed**

Hostid Types section of this form. If you leave all these checkboxes blank, the default from *rlm_isv_cfg_actpro_allowed_hostids()* will be used.

Once you have created a product definition, the list will be re-displayed, as shown above.

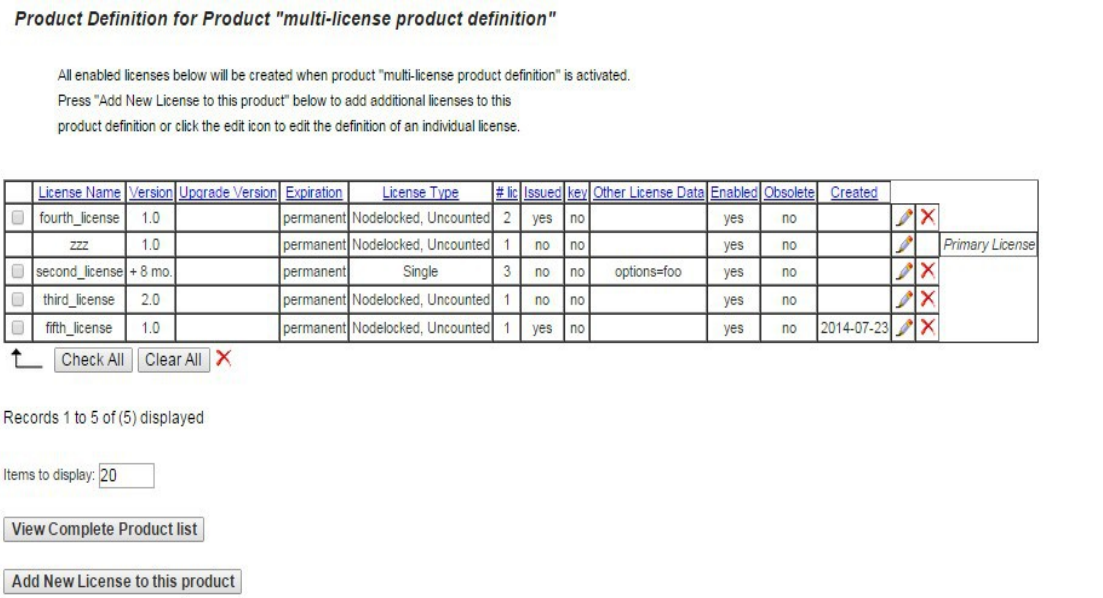
A product definition can specify one or more licenses to be created. The first license you define is called the "Primary License" for this product definition. Once you have created the Primary License, you can add additional licenses by pressing the Edit icon on the right-hand side of this list. More on this later

Multiple-license product definitions are indicated by the trailing "..." after the Primary License name, as above for the "test". "multi-license product definition", "mixed1", and "mixed2" products. ***Note: multiple-license product definitions cannot be activated with a rehostable hostid.***

If any product definition has no associated activation keys defined, you can delete it by pressing the red "X" in the far right-hand column. In this example, "another test product" has no activation keys defined, so it can be deleted. If the product has multiple licenses, it cannot be deleted from this screen. You must first edit the product by clicking on the edit icon, then in that screen, you delete all the licenses other than the primary license. At that point, the product definition can be deleted on this screen by clicking on the red X on the right.

If the version for a product is a date-based version, it will be displayed as "+ N mo.", where N is the number of months after activation.

When you click on the Edit icon (the pencil icon on the right-hand side of this screen), you will get the browser for that particular product definition, which allows you to create new licenses for that product as well as editing any of the existing licenses. The browser looks like this:



In this example, the product "multi-license product definition" has 5 associated licenses. When activated, all 5 of these licenses would be created. The primary license is the "zzz" license, and in addition, the "second_license", "third_license", "fourth_license" and "fifth_license" licenses would be created. This example contains an error: 2 of the licenses specify a "# lic" value that is

not 1. Since these are nodelocked-type licenses, this value will be ignored by the license generator. (also note that "fifth_license" was added with actpro v11.2, so it has an associated creation date).

If you want to add an additional license to this product definition, click the "Add New License to this product" button at the bottom, and you will be presented with the "Create Product" screen. In addition, you can edit the data for any of the licenses by clicking on the edit icon on the right. You can also delete any of the non-primary licenses in this screen at any time.

To return to the full list of products, click the "View Complete Product List" button at the bottom.

Once created, a product can be ***disabled*** (by unclicking on the "enabled" checkbox in the editor. If a product is not enabled, licenses will not be created for this product. This means that for a multi-license product definition, you can selectively disable individual licenses in the product definition and these licenses will not be created by the license generator. If the primary license is not enabled, then activation keys can no longer be created for this product, either.

In addition, the primary license for a product can be marked ***obsolete,*** starting in v11.3. If marked obsolete, old activation keys continue to operate, however, new activation keys cannot be created.

Activation Keys

If you are going to use RLM Activation Pro, you will then need to create activation keys - these are the keys you give to your customers. When they have the activation key, they can then use Activation Pro to fulfill licenses for their products. Activation keys can be set up to allow unlimited use, or a fixed number of uses.

Note that the single quote ('), double quote (") and semicolon (;) characters are illegal in activation keys.

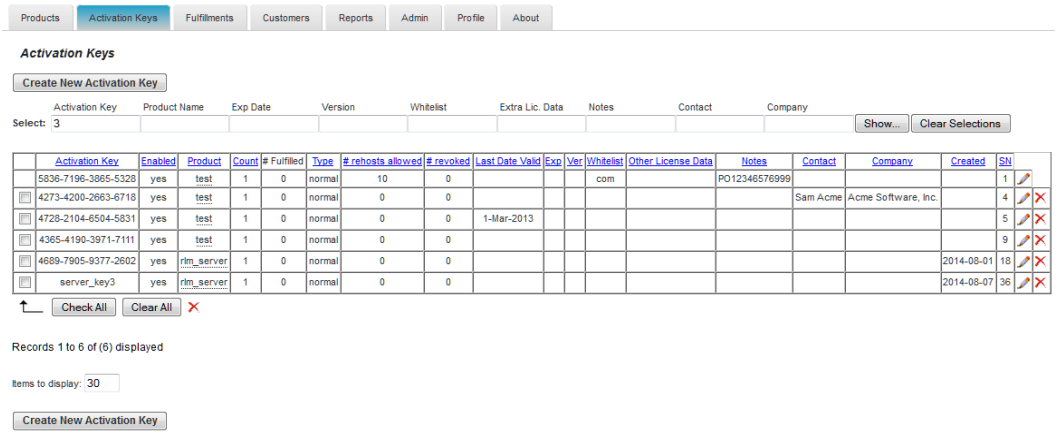
Activation keys consist of:

- an activation type
- a product definition name to be fulfilled
- the fulfillment count
- the number of independent activation keys to create
- the hostid change (rehost) count (for rehostable hostid types)

In addition, when you create activation keys, you can also specify:

- the last date valid for this activation key
- an expiration date which overrides the expiration in the product definition
- a license version which overrides the version in the product definition
- the list of allowed domains for this key
- optional additional attributes for the license
- optional notes for the activation key (unused by RLM)
- the actual activation key string
- a contact person to associate with this activation key.
- a list of allowed hostid types, if you want to override the default and/or product definition

When you select the **Activation Keys** tab you will see a list of activation keys:

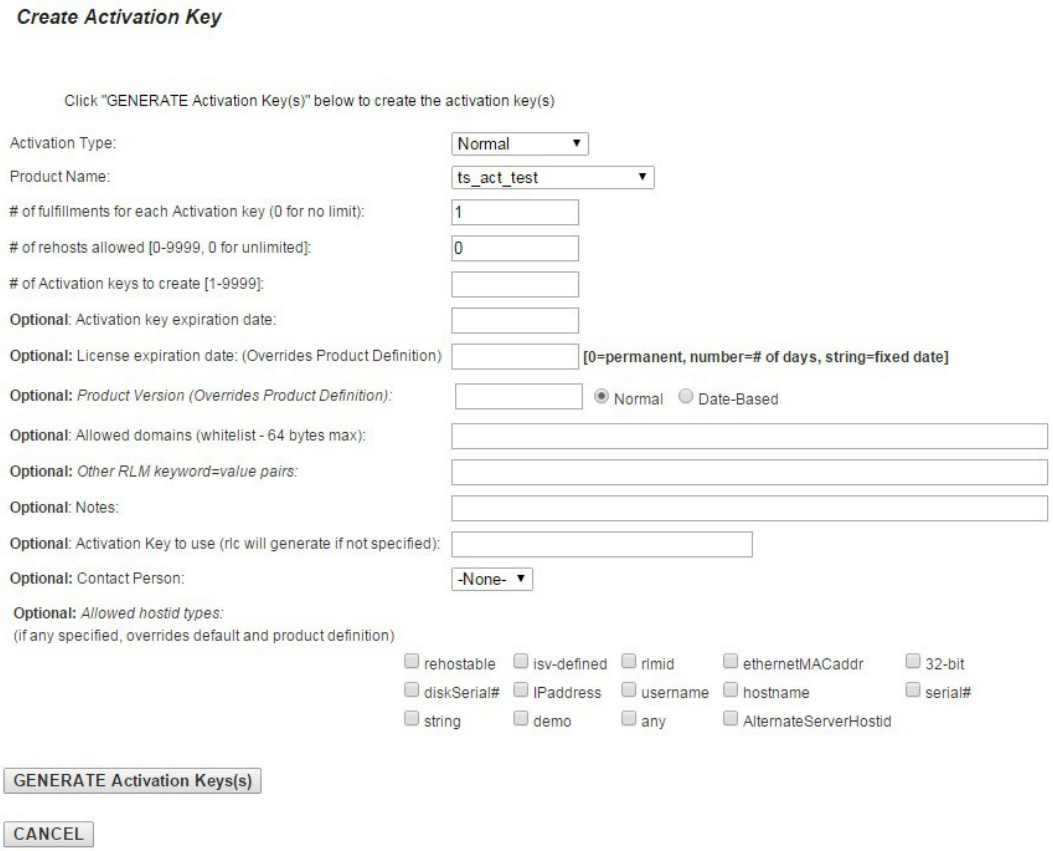


Note that in this list, only activation keys with the number "3" in the key are displayed.

At the bottom of the list there are 2 buttons ("Check All" and "Clear All"), and a red X:

Every activation key which can be deleted (ie, activation keys with the red X on the right-hand side of the list) will have a checkbox on the left. If you would like to delete multiple activation keys in a single operation, check the boxes corresponding to these activation keys then click the red X at the bottom of the form. Note that you cannot delete an activation key which has any fulfillments associated with it.

In order to create an activation key, press the "**Create New Activation Key**" button at the top (or bottom) of the screen. When you do this, you will see the "Create Activation Key" screen, as shown here:



The **Activation type** is one of the following:

- Normal
- Reactivate
- Normal-Regen (new in v11.1)
- Refresh (deprecated in RLM v9.3)
  The meaning of these 4 types is described below.

The **product name** is the name of the product definition to be activated.

The **# of fulfillments** is the number of fulfillments allowed by this activation key. For Refresh keys, this is always 1, and even if you enter a larger number, rlc stores the fulfillment count as 1. The meaning of fulfillment count depends on the type of fulfillment, see below.

The **# of rehosts allowed** is used for the revocation of rehostable licenses. Used for both Normal and ReActivate activation types, this controls the number of times your user can revoke their license and then re-activate it on another machine. If set to 0, an unlimited number of rehosts are allowed. If set to a positive number, this is the total number of rehosts that are allowed. If an attempt is made to revoke a license which has already been revoked the # of rehosts times, the application will get an RLM_ACT_TOOMANY_HOSTID_CHANGES error. (Note: prior to v11.0, the # of rehosts allowed was used for the Refresh fulfillment type only and is the number of times a hostid can be changed while this activation key is valid. After this number of hostid changes, licenses can only be refreshed for the last hostid.)

The **# of activation keys to create** controls how many activation keys are created with the data in this form. If the key string is specified (see below), this exact key will be used if the # of activation keys is 1. If the # of activation keys is > 1, then the key string will have a 16-digit string appended for each unique activation key generated.

The **Allowed domains (whitelist)** is a space-separated list of domain names which are allowed to activate using this activation key. If a request comes in from another domain, it is rejected with RLM_ACT_NOT_WHITELISTED. Note that the names in this list can be a substring of the hostname requesting the activation. So, for example, if you put the string "com" into this field, activation will be possible from any domain with "com" in the name (ie, most domains). All names are case-insensitive, and the whole list must be less than RLM_MAX_HOSTNAME (64) bytes. If an activation key specifies allowed domains, the general activation **blacklist** is not used, and the domain requesting the activation must be part of the allowed domains specification.

The **Activation Key Expiration date** for the activation key is the last date when the activation server will generate a license for this activation key. The activation server will continue to send previously-generated licenses if so requested. Note that this is completely different from the license expiration date in the product definition – the last date valid is the last date when the activation server will create a new license for this activation key. Beginning in Actpro v12.0, you can specify an expiration date as YYYY-MM-DD, for example 2016-05-23 for May 23, 2016. You can convert all expiration dates in your database to this format using the "Normalize Dates" function in the Admin/Database tab.

The **License Expiration Date**, if specified, will override the expiration date specified in the product definition. If not specified here, the license expiration from the product definition is used. The date is specified in the same way as in the product definition, i.e., a positive integer indicates a license that expires that number of days from the date of activation, and a fixed date specifies a fixed expiration date. Any expiration date specified here overrides the expiration date of *all licenses created* by the product definition. Beginning in Actpro v12.0, you can specify a fixed expiration date as YYYY-MM-DD, for example 2016-05-23 for May 23, 2016. The actpro license generator will convert this date to the standard RLM format when the license is created. You can convert all expiration dates in your database to this format using the "Normalize Dates" function in the Admin/Database tab.

The **Product Version**, if specified, will override the version specified in the product definition. If not specified here, the license version from the product definition is used. The version is specified in the same way as in the product definition, i.e., either a "normal" floating-point format version number, or an integer number of months for a date-based version. Any version specified here overrides the version of *all licenses created* by the product definition.

The **Other license parameters** are additional RLM license attributes which will be placed into the generated LICENSE. Any options specified in the activation key override corresponding attributes in the product definition, thereby allowing you to have default attributes which you modify for a particular customer when you create their activation key. These must be valid RLM license keyword/value pairs, e.g. "options=professional". If you specify other license parameters in your call to *rlm_activate()* (or *rlm_act_request()*) those parameters will override both the product definition and the activation key.

The **Notes** field is unused by the activation software, other than in listings and reports.

The **Contact Person** choicelist allows you to associate a customer with this activation key. Create customers with the "**Add Customer**" button in the Setup Commands area at the top.

If you want to override the list of allowed hostids specified in rlm_isv_config.c with the *rlm_isv_cfg_actpro_allowed_hostids()* call (or the product definition, if specified there), you can select a set of allowed hostids in the **Allowed Hostid Types** section of this form. If you leave all these checkboxes blank, the default from *rlm_isv_cfg_actpro_allowed_hostids()* or the product definition will be used.

The **activation key** generated by **rlc** is a 16-digit number (with embedded dashes for a total of 19 characters) of the form:

aaaa-bbbb-cccc-dddd

for example:

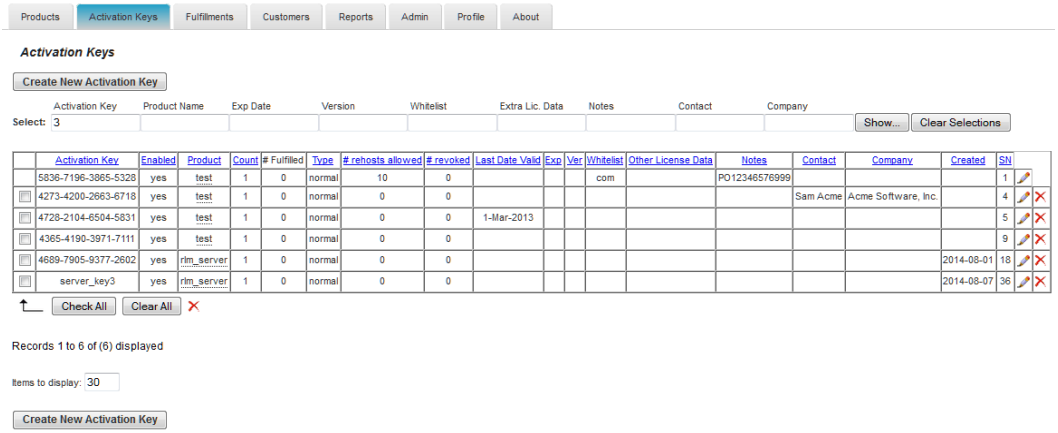
4567-4997-7125-5436

Alternately, **rlc** will allow you to specify the activation key. If you specify a key and generate a single key, the exact key you specify will be used. If you generate multiple keys, **rlc** will append 16 characters (with 4 dashes) to the string you specified.

This activation key is treated as a string by all RLM activation components.

When you have completed entering the data, press the "**Generate Activation Key(s)**" button at the bottom of the form. Once you do this, you will see a screen with the activation key name(s) displayed, and a "Back to List" button. Pressing this button re-displays the list of activation keys.

This list gives you an overview of the activation keys in your Activation Pro system. If any activation key has no associated fulfillments, you can delete it by pressing the red "X" in the far right-hand column. In this example, none of the keys have any fulfillments, so they can all be deleted. Note that this list was once again filtered to display only activation keys which contain the string "3".



While viewing activation keys, hovering the cursor above the product name will display a popup listing the primary product's license name, version, and type.

You can also view the fulfillments done for a particular activation key by clicking on the "Show" icon (a small image of a computer screen) at the end of each row in the **Activation Keys** display. (The display above has no fulfilled licenses, so each activation key displays the red-X delete icon rather than the show fulfillment icon).

Activation Keys which are disabled (not deleted) will appear in gray in the list, and can be re-enabled later.

Fulfillment Type

The **type of fulfillment** controls how the activation server processes requests. In all cases, for a node-locked license, the fulfillment count field represents the total number of independent licenses which can be generated. For floating licenses, the fulfillment count field represents the total license count of all floating licenses generated (multiplied by the individual license "# of licenses" multipliers, of course).

The **Normal** fulfillment type is the most straightforward activation type, and represents what most people think of as software activation: each request from a new hostid generates a new license,

and this license's expiration date is computed at the time the license is activated. A subsequent request from the same hostid will cause the old generated license to be retrieved, with exactly the same license parameters. If you use the **Normal** fulfillment type for Rehostable licenses, you should always use a fixed expiration date, not a # of days, since if the user revokes the first license and re-activates it, a new license will be created, and you probably don't want to allow the expiration date to move past the original expiration date.

The **Normal-Regen** fulfillment type works just like the **Normal** fulfillment type, *except that new activation requests from the same hostid generate a new license, and consume activation count.* This means that your user can re-activate the license on the same hostid and get a different license. ***If you use this for any kind of counted license (floating or node-locked, counted), this means the customer can get N times the number of licenses you have authorized, SO PLEASE USE THIS ACTIVATION TYPE WITH CAUTION.*** If you use the **Normal-Regen** fulfillment type for Rehostable licenses, you should always use a fixed expiration date, not a # of days, since if the user revokes the first license and re-activates it, a new license will be created, and you probably don't want to allow the expiration date to move past the original expiration date.

Why would you want to use a **Normal-Regen** fulfillment type? The main reason would be for licenses where you do not really want to count how many fulfillments are done, but that you do want to change other parameters in the activation key so that your customer can re-activate the license using the same activation key and get their new license. With a **Normal** fulfillment, the customer would get the original license, which isn't quite so useful. As an example, licenses for RLM itself are not counted, but they are locked to an ISV name with the customer= field, and they specify the licensed RLM platforms using the platforms= field. We give activation keys to our customers for RLM, however, often a customer adds a platform. When this happens, we modify the platforms= definition in the activation key, and our customer can re-activate using the same activation key. (We also modify the version in the product definition when we release a new version of RLM.) We use a **Normal-Regen** key type for this. Another example is our fulfillments for Activation Pro itself. We do not count the number of activation servers you run, but we do issue the licenses for one year. When you renew support, we update the expiration date in the activation key, and you are able to retrieve your new license with the same activation key.

For the **Reactivate** fulfillment type, additional activations on the same activation key are considered reactivations. A reactivation will have the same expiration date as the original activation. This can be used to allow your customer to move a time-limited license to a new host one or more times. So, for example, assume you sell a 1-year license, but you would like your customer to be able to move it one time during the year. In this case, set the fulfillment count to 2 and the activation type to **Reactivation**. Then, once your customer creates the first license it will have an expiration date one year out, but he can come back sometime later during that year and reactivate the license on another system. The new license would expire on the original expiration date. If he waited more than one year, no new valid license could be generated. You should use the **Reactivate** fulfillment type for Rehostable licenses if you want to deliver a license which expires some number of days after it is first activated. In the case of rehostable licenses, however, the fulfillment count will NOT control how many times they can revoke and re-activate the rehostable license, and should be set to the actual fulfillment count you desire, since a revocation of a rehostable hostid returns activation count to the activation key. Beginning in v11.0, the # *rehosts* parameter will control how many times your user can revoke the rehostable license and re-create it. If you set the fulfillment count to a number > 1, you should be sure to set the # *rehosts* high enough to accommodate all licenses which will be activated with that particular activation key.

Refresh-type activations were deprecated in RLM v9.3.

This processing for each type of activation key is described in the following table:

Activation type	New request from the same hostid (with same count)	New request from new hostid (or same hostid with different count)	expiration date of license
Normal	prior license returned	new license generated if sufficient fulfillment count is available	computed at activation time
Normal-Regen	New (different) license generated and returned if sufficient fulfillment count is available.	new license generated if sufficient fulfillment count is available	computed at activation time
Reactivate	prior license returned	new license generated	same expiration date as first activation done with this activation key
Refresh (deprecated in RLM v9.3) – use REHOSTable licenses instead.	new license generated if request is on a new day, otherwise prior license returned	new license generated until hostid change count exceeded, then RLM_ACT_TOOMANY_HOSTID_CHANGES error returned	computed at activation time

In the table above, whenever the action listed is "prior license returned", the remaining fulfillment count is not decremented. Whenever activation generates a new license, the remaining fulfillment count is decremented by the license count generated . If a rehostable hostid is revoked, the fulfillment count is incremented.

Other RLC Commands

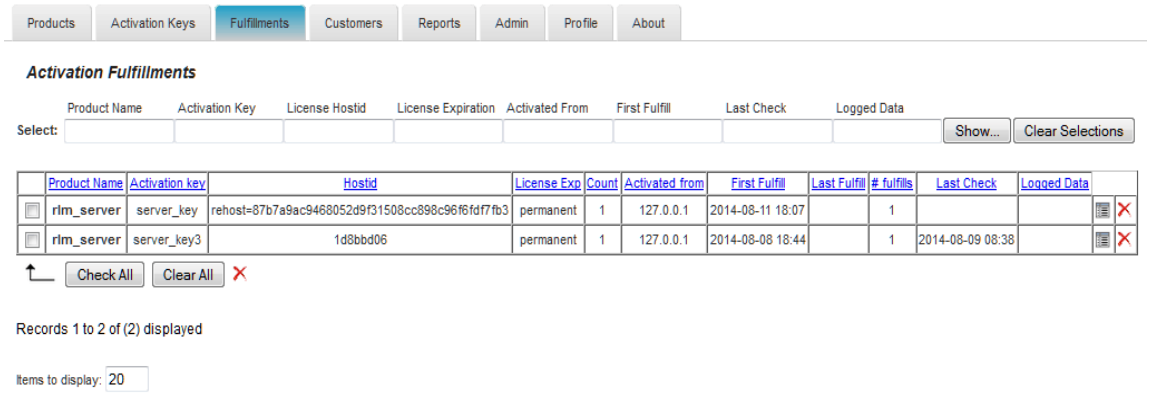
Once you have set up a product definition and an activation key, you can begin using Activation Pro, as soon as you upload your license generator settings file and your Activation Pro license, as described in the sections below. There are, however, a number of other RLC commands in the “Fulfillments”, “Customers”, “Reports”, “Admin”, “Profile”, and “About” tabs which will be useful in the day-to-day operation of the system, so you will want to become familiar with them before you enable Activation Pro for your customers.

Fulfillments

The **Fulfillments** tab allows you to view the licenses which have been created by Activation Pro. In addition, you can delete an individual fulfillment, in order to re-enable the count consumed on that activation key by that particular fulfillment. This is helpful when you are testing the activation software and you want to be able to regenerate a license for a particular hostid, since a re-activation from the same hostid would return the old license. By deleting the old fulfillment, the activation software will generate a new license (with a new expiration date) for that activation key on the hostid. The Expiration Date displayed in the browser will be the earliest expiration date for all licenses in the case of a multiple-license product definition.

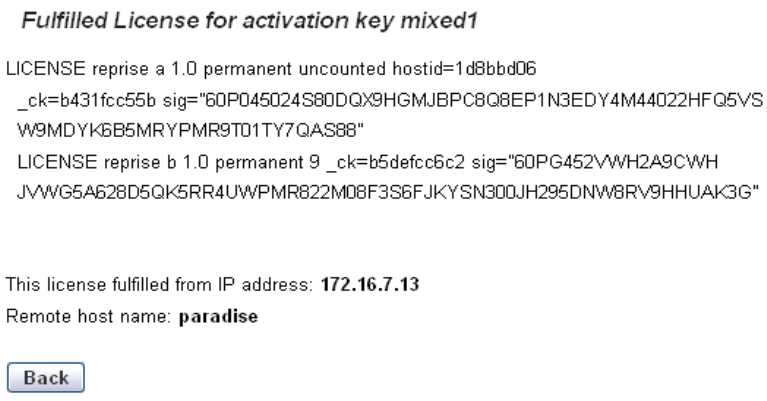
The "Last Check" column indicates the last time rlm_act_keyvalid() was called on this activation key/hostid combination. (This column is new in Actpro v11.2BL2).

As in all the lists, you can filter the list using the Select: box at the top:



To delete the fulfillment, click on the red X at the right-hand side. Alternately, at the bottom of the list there are 2 buttons ("Check All" and "Clear All"), and a red X. Every fulfillment which can be deleted (ie, fulfillments with the red X on the right-hand side of the list) will have a checkbox on the left. If you would like to delete multiple fulfillments in a single operation, check the boxes corresponding to these fulfillments then click the red X at the bottom of the form.

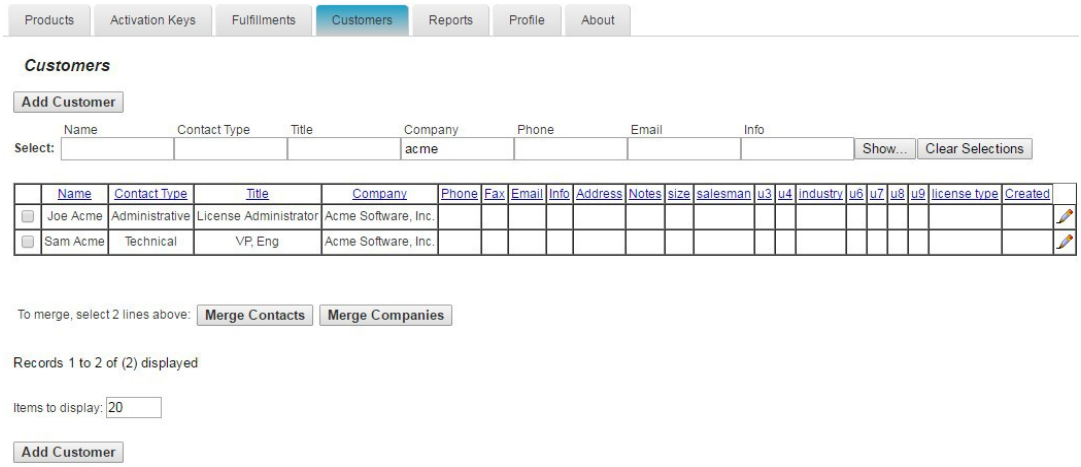
To view the fulfillment, click on the list icon on the right. Clicking the list icon displays the fullfillment data:



Customer List

RLM Activation Pro allows you to maintain a customer list and associate customers with activation keys.

You manage customers by selecting the “Customers” tab, as shown here:



Editing Customer Information

You can edit individual contacts with the edit button on the right. If the contact is not associated with any activation key, pressing the delete button will confirm that you want to delete the contact. If you delete the last contact for a company, you will be prompted to delete the company record itself.

Merging Customers

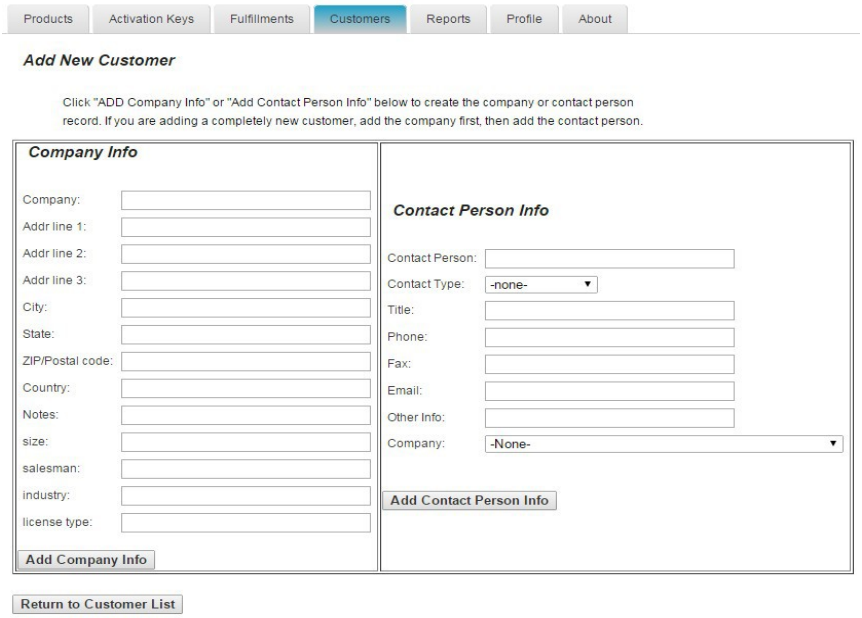
If you have 2 contacts who represent the same person, or 2 companies which are the same, you can check the checkboxes at the beginning of the 2 lines, then press “***Merge Contacts***” or “***Merge Companies***”, as appropriate. You must select exactly 2 customers for the merge operation. Once you press the “Merge...” button, you will be prompted to select which of the 2 should remain after the merge operation.

A merge of contacts will update any activation key or user record which referred to the old contact to then refer to the new (surviving) contact. For a merge of companies, any contact associated with the old company will now be associated with the surviving company.

Once the merge is complete, in the case of Merge Companies, the other company will be deleted, and all data associated contained in company record will be lost. For the case of contacts, you will be prompted as to whether you want to delete the contact or not. If you delete it, all information in that contact record will be lost.

Creating new Customers

To create a new contact/customer, press the “**Add Customer**” button at the top or bottom of the screen. When you press the “**Add Customer**” button, the “Add New Customer” form appears:



You add the company first, then add contact people for that company. Note that you are not required to add company information for contacts if you prefer to have only the person's name and other info from the Contact Person Info side of the form, however, the Activation Pro customer portal operates using Company Information, so if you plan to use the customer portal, you should add company names along with your contacts.

Once you have added some Contacts to the system, you can assign them to activation keys.

After creating contacts, the list is displayed, as above.

Reports

Activation Pro has a report generator, which you access via the "Reports" tab. The report generator is described in detail in the Reporting chapter on page 48.

Admin

The admin tab, available to a user with "admin" access, provides access to a second level of tabs to perform a number of administration functions. Each of these functions is described in the sections which follow.

ActPro License

The license generator in Activation Pro requires a license from Reprise Software to operate. When you purchase Activation Pro, Reprise sends this license to you. To install the license, select the "ActPro License" tab, lick the "**Browse...**" button to locate your license file, then press "**Upload File**". Your license file will be uploaded to the website.

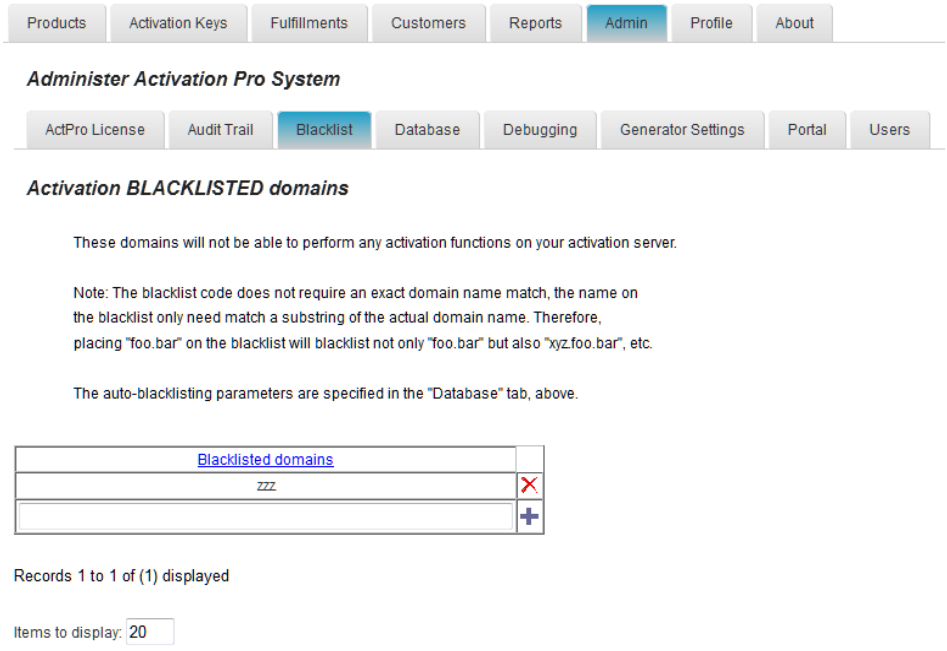
Audit Trail

All create/edit/delete operations on product definitions, activation keys, customer and contact data, as well as deletions of blacklist and license fulfillments are recorded in the ActPro *Audit Trail*. Each entry in the audit trail lists the time of the operation, who performed it, the operation (add/edit/delete), the database table and the key in that table.

The *Audit Trail* tab allows you to view the audit trail as well as clear old entries.

Blacklist

RLM Activation allows you to specify a list of unauthorized domains - domains which cannot activate licenses. If you select the "Blacklist" tab in the Admin tab area, you will see a list of blacklisted domains. The last item in the list is blank and allows you to add a new domain. In addition, each domain in the list has a **Delete** button on the right to remove it from the list:



The blacklist is a list of domain names which are not allowed to activate using this activation key. If a request comes in from any domain on this list, it is rejected with RLM_ACT_BLACKLISTED. Note that if any string you enter here is a substring of the domain name then that domain will be rejected. All names are case-insensitive, and each item in the list must be less than RLM_MAX_HOSTNAME (64) bytes. If an activation key specifies allowed domains, the **blacklist** is not used, and the domain requesting the activation must be part of the allowed domains specification in the activation key itself.

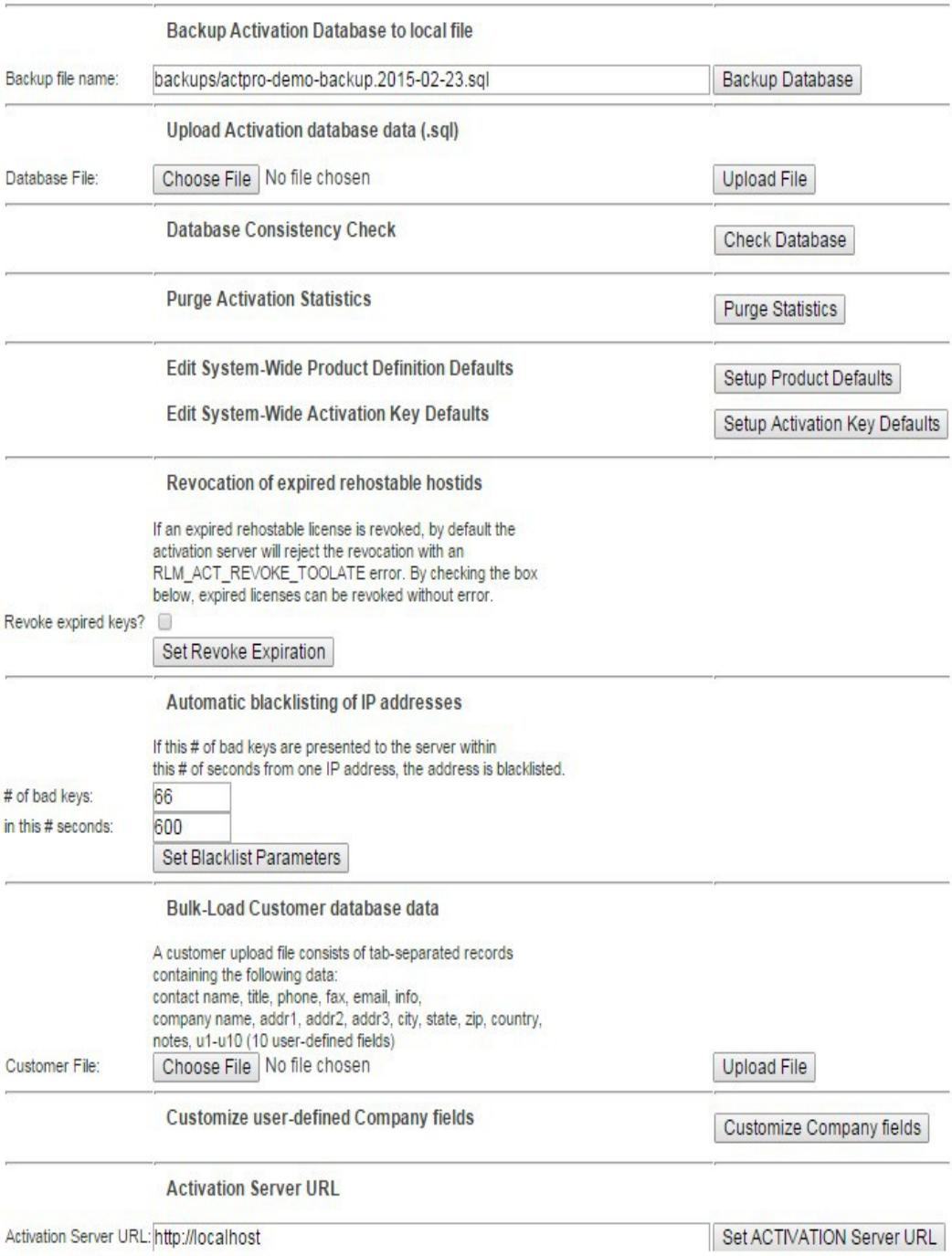
Beginning in RLM v9.4, the license generator automatically blacklists IP addresses which attempt a large number of incorrect activation keys. These ip addresses will appear in normal IP address format (a.b.c.d), rather than as domain names, and an exact IP address match is required to blacklist the address, rather than the substring match that happens with manually-blacklisted domain names. When you are manually entering blacklisted domains, do *not* enter them in IP address format. You can delete any automatically-blacklisted IP address in this list by pressing the delete icon on the right-hand side of the list. For more information on the automatic blacklisting of IP address see Appendix K – Automatically Blacklisting IP addresses on page 96.

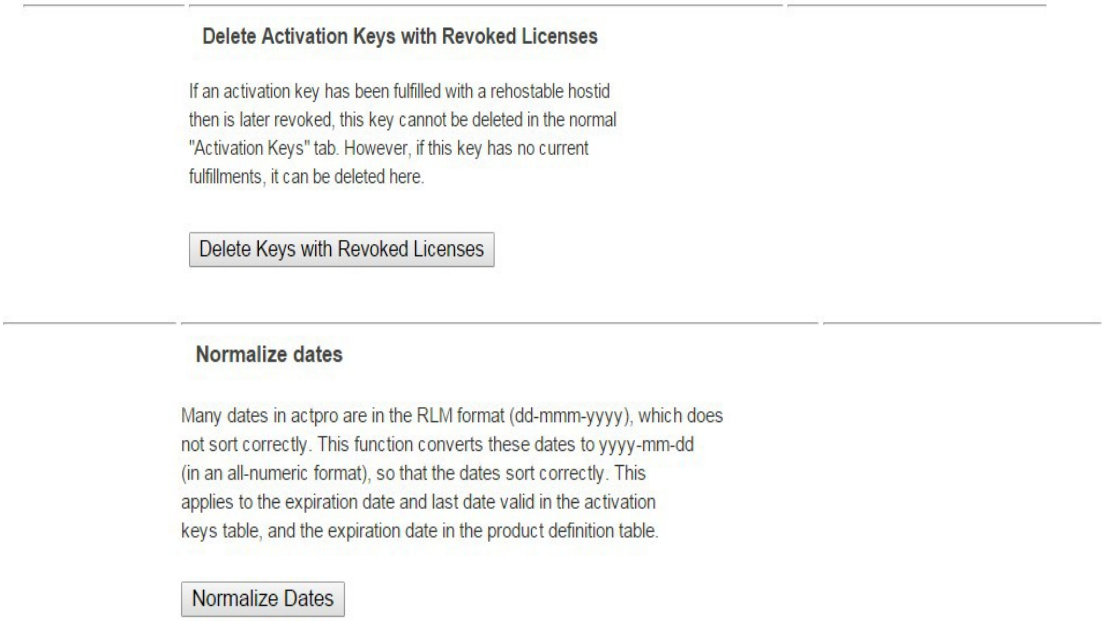
Database Tab

The "Database" tab selects a page of various database-oriented operations. On this page, you can:

- backup your database,
- restore a previously-backed-up database (or just a part of a database),
- perform a consistency check on the database,
- purge activation statistics,
- control whether revoked expired rehostable hostids return count to the activation key,
- set the automatic blacklisting parameters, and
- bulk-load customer data, and customize the customer data fields.
- Delete activation keys which have previously been revoked from a rehostable hostid. (These keys cannot be deleted in the normal "Activation Keys" tab.)
- Normalize the dates in the database.

The contents of the database tab are shown here:


Backup Activation Database to local file
Backup file name:
backups/actpro-demo-backup.2015-02-23.sql
Backup Database
Upload Activation database data (.sql)
Database File:
Choose File
No file chosen
Upload File
Database Consistency Check
Check Database
Purge Activation Statistics
Purge Statistics
Edit System-Wide Product Definition Defaults
Setup Product Defaults
Edit System-Wide Activation Key Defaults
Setup Activation Key Defaults
Revocation of expired rehostable hostids
If an expired rehostable license is revoked, by default the activation server will reject the revocation with an RLM_ACT_REVOKE_TOOLATE error. By checking the box below, expired licenses can be revoked without error.
Revoke expired keys?
Set Revoke Expiration
Automatic blacklisting of IP addresses
If this # of bad keys are presented to the server within this # of seconds from one IP address, the address is blacklisted.
# of bad keys:
66
in this # seconds:
600
Set Blacklist Parameters
Bulk-Load Customer database data
A customer upload file consists of tab-separated records containing the following data:
contact name, title, phone, fax, email, info,
company name, addr1, addr2, addr3, city, state, zip, country,
notes, u1-u10 (10 user-defined fields)
Customer File:
Choose File
No file chosen
Upload File
Customize user-defined Company fields
Customize Company fields
Activation Server URL
Activation Server URL:
http://localhost
Set ACTIVATION Server URL



Debugging Tab

The Activation Pro license generator can write debug log information to the database. This is disabled by default, since the amount of data can become quite large, and it's generally only useful for diagnosing particular activation-related problems. Selecting the "Debugging" tab displays a page with 3 buttons: "**View Debug Log**", "**Enable Debug Logging**", and "**Disable Debug Logging**". If you are experiencing a problem with activations, you can go to this page, enable the debug logging, then re-try the activation and view the debug log. In addition, starting in v12, you can select (filter) the debug log using the "Select" box on the "View Debug Log" line.

While viewing the debug log, you can clear the log, and refresh the log while on the page.

Generator Settings Tab

The RLM Activation Pro license generator is generic, and it is customized via a small settings file (similar to the ISV server settings file in RLM). This generator settings file, however, ***contains your company private key,*** so you need to be very careful with it. Before you can create licenses with Activation Pro, you need to upload the Generator Settings File to the website. To do this, select the "Generator Settings" tab, click the "**Browse...**" button to locate your settings file, then press "**Upload File**". Your generator settings file will be uploaded to the website.

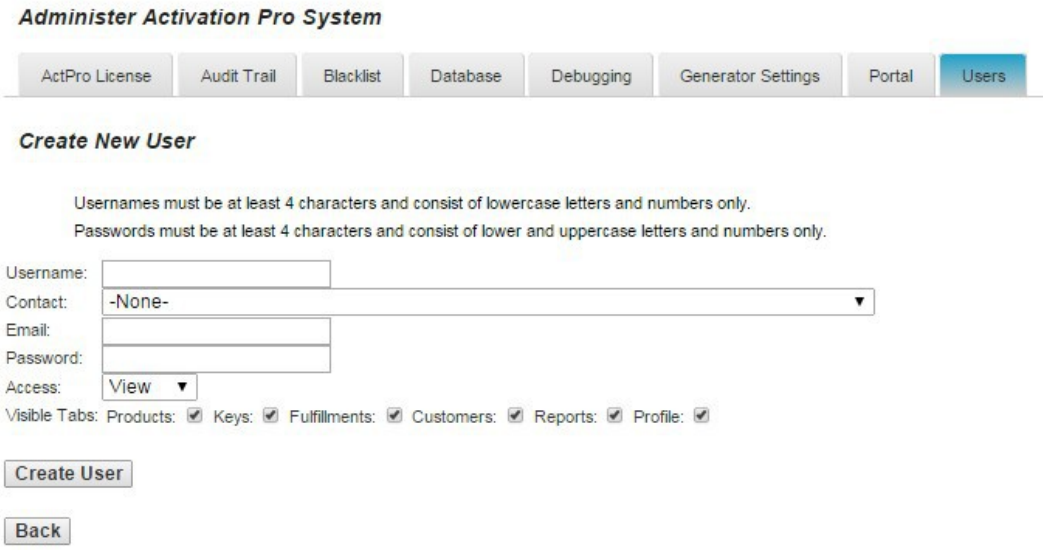
This tab also allows you to control whether requests to the activation server must be encrypted or not. Prior to Actpro v12.0, requests could be encrypted or unencrypted. The RLM web interface, for example, uses unencrypted requests, as does any web page you set up to do activation, whereas the rlm_activate() and rlm_act_request() calls always encrypt the request. The disadvantage of allowing an unencrypted request is that any user can set up a web page to access your activation server and pass data which you might not want to allow (such as values for optional RLM parameters.) Actpro v12 defaults to allowing unencrypted requests, but if you use optional parameters, you might want to turn these off.

Portal

The “portal” tab allows an administrator to set the (relative) directory to the portal files, set the portal name, install the portal software, and upload a logo to be used in the portal. More details on the customer portal are contained in the Customer Portal chapter on page 51.

Users

The “users” tab allows an administrator to view the list of authorized users, add new users, delete users, and edit an existing user. The user creation screen is shown here:



At the bottom of the form is a list of checkboxes labeled “Visible Tabs”. By default, they are all checked. If you want to make a particular tab disappear for a user, uncheck the checkbox next to the tab name. For example, if you uncheck the “Profile” checkbox, then the “Profile” tab will not appear for this user, which means that the user cannot change their password or set defaults for product definitions or activation keys. Note that the “Admin” and “About” tabs cannot be hidden.

Profile

Clicking on the “Profile” tab allows you to edit the email address associated with your account, or change your password.

About

Clicking on the “About” tab displays a screen of information about this Activation Pro installation, including the expiration date of your Activation Pro license (Note that the expiration date is updated by the license generator, so if the license generator has never run, this information will not appear. Also, the date won't be updated when you install a new license until the license generator runs again). In addition, there is a link to the RLM Activation Pro Manual on this page:

RLM License Center (RLM Activation Pro), v11.3

This web interface is used to administer the *RLM Activation Pro database*.
rlc allows you to perform status and administration functions on the RLM activation database.

To begin, please log in. Then choose a tab at the top.

ISV:	**demo**
Server host:	**localhost**
Database Server host:	**127.0.0.1**
Database:	**RLM_Activation_Pro_test**
RLM Activation Pro version:	**v11.3BL1-p0 (13-feb-2015)**
Current Local Server Time:	**12:57 Mon, 2015-02-16**
License Generator Expiration:	**1-jan-2014**

Enabling Your Application for Activation

The last step in setting up Activation Pro is adding activation requests to your application, as described in this chapter.

On the client side, you must prepare your activation code and get the activation key to your customer.

Making the activation request for your customer - using the activation API

Once you are familiar with how to use the activation server and the activation GUI, you are ready to integrate the activation request into your application. To do this, embed a call to rlm_activate() at the appropriate point (usually after attempting to check out a license which fails), and, if you are successful at activating the license, perform the following steps:

- call rlm_close() on your handle
- write out the activated license to a local license filename of your choice (Reprise Software recommends a file named *something*.lic)
- call rlm_init(), and
- re-attempt the license checkout.

For details on the activation API request call, see Appendix A – Activation API on page 57.

Making the activation request for your customer - using the activation GUI

You can get started right away by using the **activation GUI** supplied by Reprise Software. We recommend that you start this way, so that you can become familiar with the server side of activation without having to embed any code into your application.

In order to proceed, make sure that you have set up the server side as described in Activation Pro Setup on page 15 and created at least one product definition and activation key as described in the Product Definitions and Activation Keys sections of Your Activation Database, starting on page 21

The activation GUI is built into the rlm webserver. Run rlm (you don't need a license file), point your browser at port 5054 (or the port number you specified in the -ws port# specification) then select "activate license" from the menu on the left-hand side. The GUI will guide you through the steps in activating a license. (Note: prior to RLM v6.0, the default admin port was 9000, not 5054).

When the rlm webserver activates a license, the resulting license file will be placed in the directory from which rlm was started, and the file name will be "activateN.lic", where N is a sequence number. Each activation performed will get a new number.

Note that RLM activation will only operate with non-zero RLM_HOSTID_32BIT, RLM_HOSTID_ETHER, or ISV-defined hostid types - all other hostid types will return an error.

Activation API or rlm web interface - which to use?

The rlm web interface is completely generic, and you can activate licenses for your products using this interface by setting up activation on your web server and specifying your URL in step 1. However, it will ***always*** be simpler for your customer if you integrate the RLM activation API into your product or installation tools, since you can preset many of the activation parameters, such as URL, ISV name, etc. Reprise Software strongly recommends that you integrate the API into your product.

Other Client-side activation options

Alternately, you can create a custom stand-alone activation utility for your customers, rather than building it into your application. Follow the steps above for using the Activation API.

Also, you can create an html page to activate licenses using your activation server. For an example of this, see “activation_example.html” in the RLM kit “examples” directory. This technique, however, will require your customer to cut and paste the resulting license file into the appropriate file on their system, so it is inferior to integrating the activation API into your application.

Using Proxy Servers

The RLM Activation client software (either the rlm_activate()/rlm_act_request() API call or the RLM admin web interface) will make use of a user-defined proxy server. In order to use a proxy server, set the environment variable **HTTP_PROXY** or **http_proxy** to the hostname and port number of the proxy server.

For example, to utilize the HTTP proxy server running on "myproxyhost" on port 8765, use the following command on unix:

% setenv HTTP_PROXY myproxyhost:8765

If your proxy server uses authentication, you use the **HTTP_PROXY_CREDENTIALS** environment variable to pass the credentials to the proxy server:

HTTP_PROXY_CREDENTIALS - the username and password to authenticate you to the proxy server, in the format user:password.
For example, if your username is "joe" and password is "joes_password":

% setenv HTTP_PROXY_CREDENTIALS joe:joes_password

Note that RLM activation supports only the BASIC authentication type.

You can either set these environment variables before running your application, or use putenv() (or rlm_putenv()) to set them inside your application before calling *rlm_act_request()*.

Proxy Auto-Detection

Beginning in RLM v11.0, RLM will attempt to auto-detect the proxy server on Windows systems. If auto-detection fails, you can set the RLM_PROXY_DEBUG environment variable to get diagnostic information written to stdout. You can also replace RLM's default auto-proxy detection on Windows, or add your own detection on other platforms. To do this, replace the

rlm_get_proxy.obj or rlm_get_proxy.o module in the library. This module contains the single entry point *rlm_get_proxy()*. *rlm_get_proxy()* is called as follows:

void *rlm_get_proxy*(const char *url, char *proxy)

rlm_get_proxy() attempts to locate the proxy server for the specified URL. If a proxy is located, the return string "proxy" is filled in with the hostname and port in the format: hostname:port

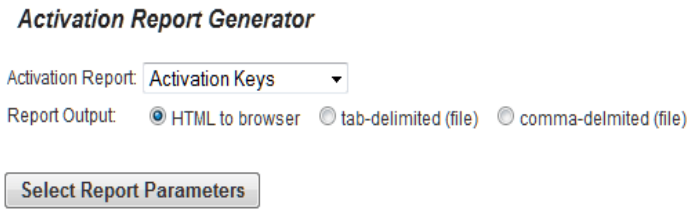
If there is no proxy, or the proxy is bypassed, rlm_get_proxy() should return an empty string.

The proxy name is allocated by the caller, and is a string of 1000 characters.

Reporting

rlc contains a reporting module which allows you to generate reports of products, activation keys, fulfillments, etc.

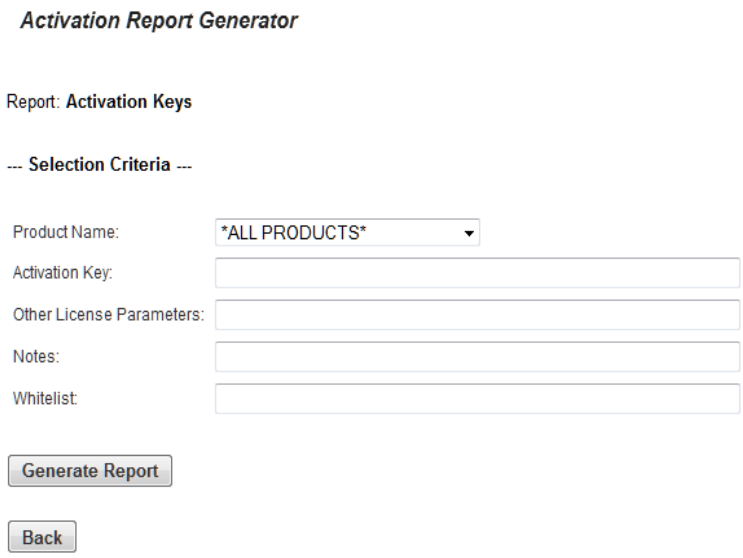
To use the reporting module, select the "Reports" tab. When you select "Reports", the first Reporting screen appears:



The first choicelist allows you to select the report type: Activation Keys, Unfulfilled Keys, Fulfillment Summary, Fulfilled Licenses, Products, Permanent Licenses, Revoked Licenses, or Statistics.

On the second line, you select the destination for the report – either directly to the browser, or to a tab- or comma-delimited file which can be imported to a spreadsheet.

After making these selections, click "Select Report Parameters" which will present a screen allowing you to filter the output of the report. The contents of this screen are report-type dependent. The selection screen for the "Activation Keys" report is shown below:



The "Product Name" choicelist allows you to select an individual product or all products.

The other text boxes allow you to filter the various fields in the activation key table.

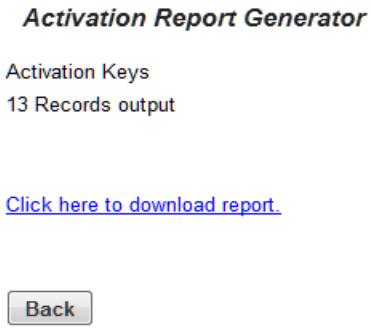
In each text box, the text you type will be used as a regular expression to select the appropriate rows from the activation key table. So, for example, if you have an activation key which is "1234-5678-abcd-efgh", then typing "1234" in the Activation Key field will select this key (as well as any others with "1234" in the activation key string) for the report.

For the "statistics" report, you can select the date to report. Enter the date as yyyy-mm-dd. If you enter only yyyy-mm, a report for that month will be produced.

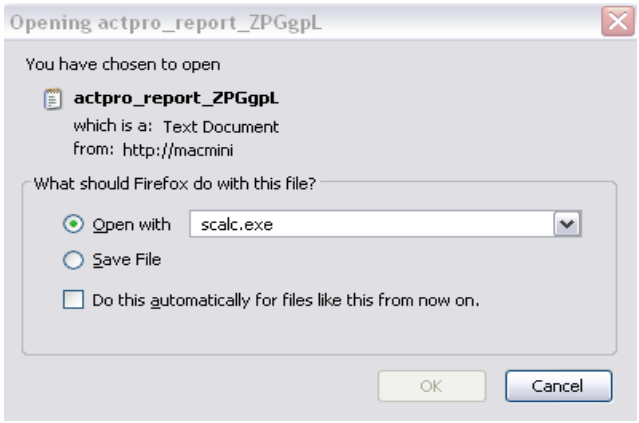
Once you have made your selection, click "Generate Report", and the report will be generated.

If you selected report output to the screen, you will see a form very similar to the "View Product Definitions" or "View Activation Keys" forms. You can page through the various pages as well as sort on the columns by clicking on the underlined column header.

If you selected either tab- or comma-delimited output, you will see the a screen similar to the following:



When you click on the underlined "Click here to download report", your browser will ask you if you want to save the file, with a dialog similar to the following (Firefox shown here):



You can either open the file directly with your spreadsheet, or save the file. Once opened, the standard spreadsheet import functions for tab- or comma-delimited files will work to put your report data into separate columns.

Release Checklist

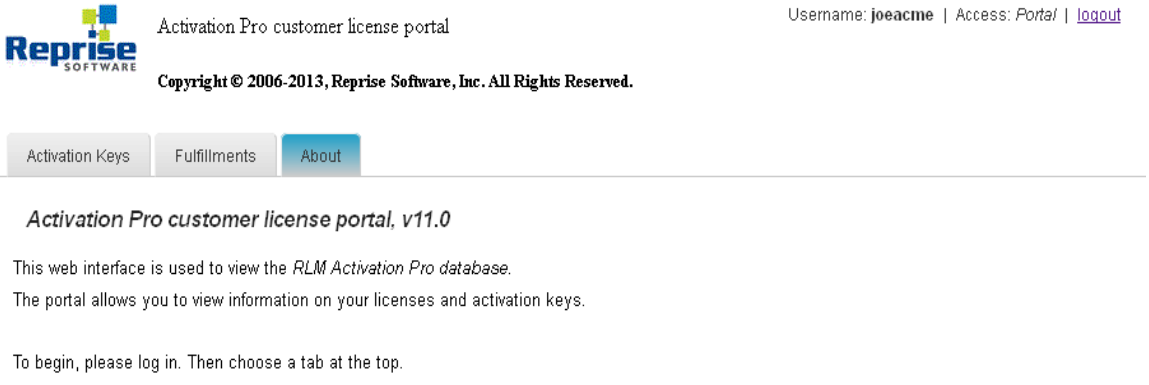
With RLM Activation Pro, you control the activation server, so you can make changes at any time. However, there are a couple of issues that you need to consider before you ship your application:

- Review the *rkn_activate()* (or *rlm_act_request()*) call you make in your application to be sure that you have the correct URL for your activation server.
- Be sure that your activation license generator and data files are protected on your webserver by limiting user access to **rlc**, especially **Admin** access.
- Be sure that your license generator (in the cgi-bin directory) , including any isv generator settings file (isv.gen) is read-write-execute to the web user only.

Customer Portal

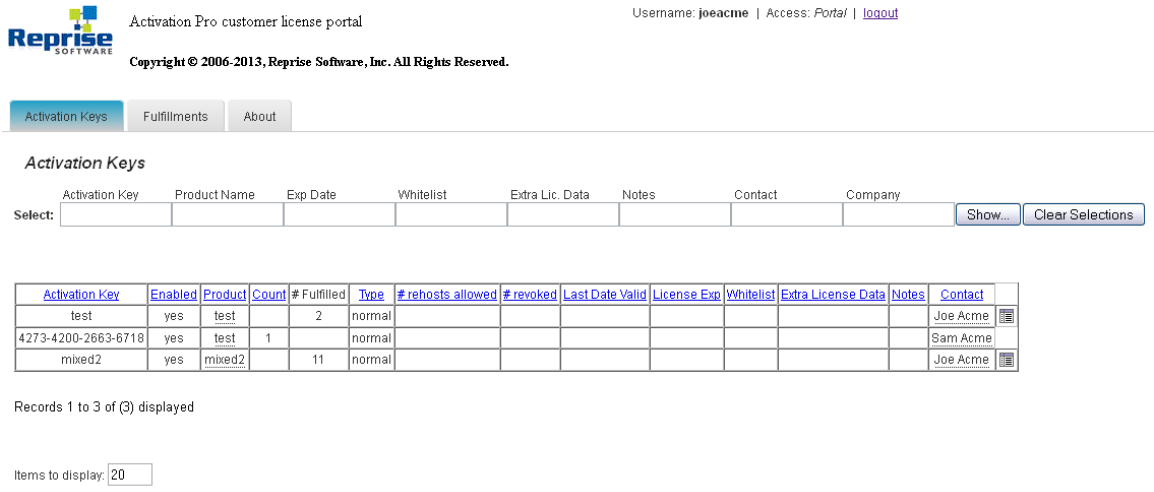
In addition to the rlc administration program, Activation Pro includes a Customer Portal to allow your customers to view their own activation keys and fulfillments. The customer portal contains a subset of the rlc functionality, specifically the ability to view activation keys and license fulfillments. In all cases, only the activation keys and fulfillments associated with the *Company* of the logged-in user are displayed.

The main customer portal screen is shown here:



As you can see, the interface is similar to rlc, but with only the "Activation Keys", "Fulfillments", and "About" tabs. Also note that only users with "Portal" access can log in to the portal. It is important that the "Portal" user has an associated contact which itself is associated with the *Company* of the user who is the contact on the activation key.

Selecting "Activation Keys", all the activation keys assigned to contacts in the company (in this case, Acme Software, Inc), are displayed:

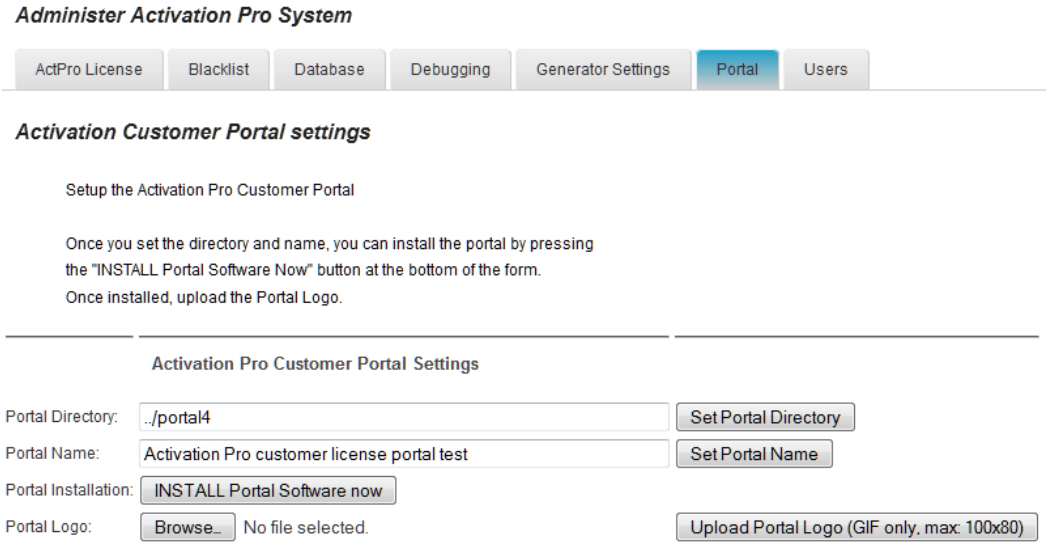


Your customer can filter the list using the Select: box at the top.

Selecting the "Fulfillments" tab allows your customer to view the licenses which have been fulfilled.

Setting up the Customer Portal

After you install Activation Pro, you can set up the portal via the Admin/Portal tab:



On this form, enter the directory as a relative path to the *actpro* directory, for example, specifying "../portal" would create the directory *portal* alongside the *actpro* directory. The "INSTALL Portal Software now" button will then install the customer portal files in this directory.

Once set up, you can customize both the title and the logo on the customer portal. To do this, set the Portal Name and press the "Set Portal Name" button, and Browse for the logo and press the "Upload Portal Logo" button. You can set the portal name any time, but you should upload the portal Logo after you have installed the portal software.

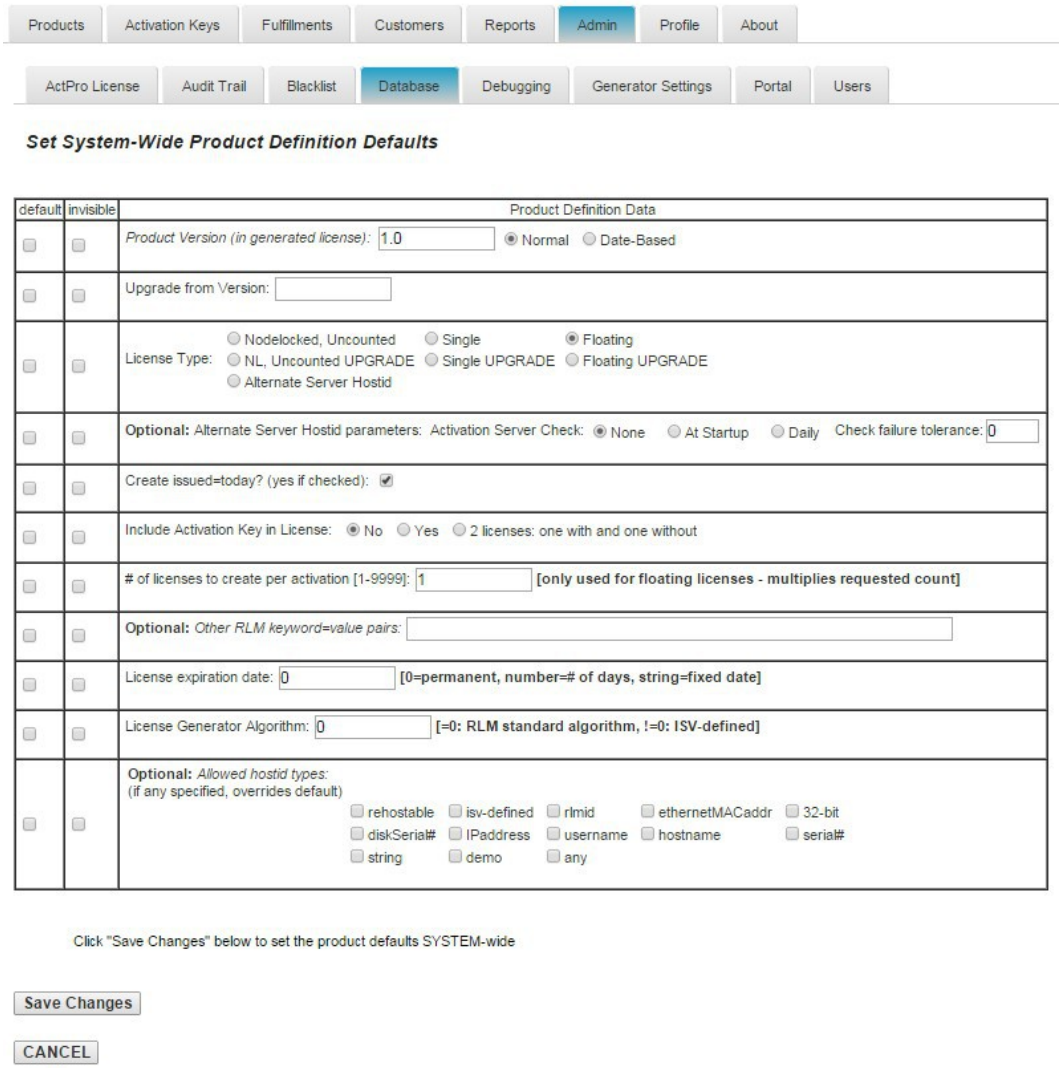
Note: uploading the logo to the customer portal depends on the portal files being on the same system as the rlc (actpro) files. Once you have updated the logo, you can move the portal files to another system outside your firewall, *as long as that system has access to your Activation Pro MySQL database server system.*

Setting Defaults for Products and Activation Keys

Activation Pro allows you to set default values for most fields in the product creation/edit form as well as the activation key creation/edit form. In addition to setting default values, a field can be marked as "invisible", which means that it will not appear in the create/edit form so it will always have the default value when created, and the value cannot be changed when edited.

There are 2 levels of defaults – system-wide defaults (edited by an administrator, with the admin->database->Edit System Wide Product Definition Defaults, or Activation Key Defaults) and user defaults (edited by the user with the profile->Edit Product Definition Defaults or Edit Activation Key Defaults).

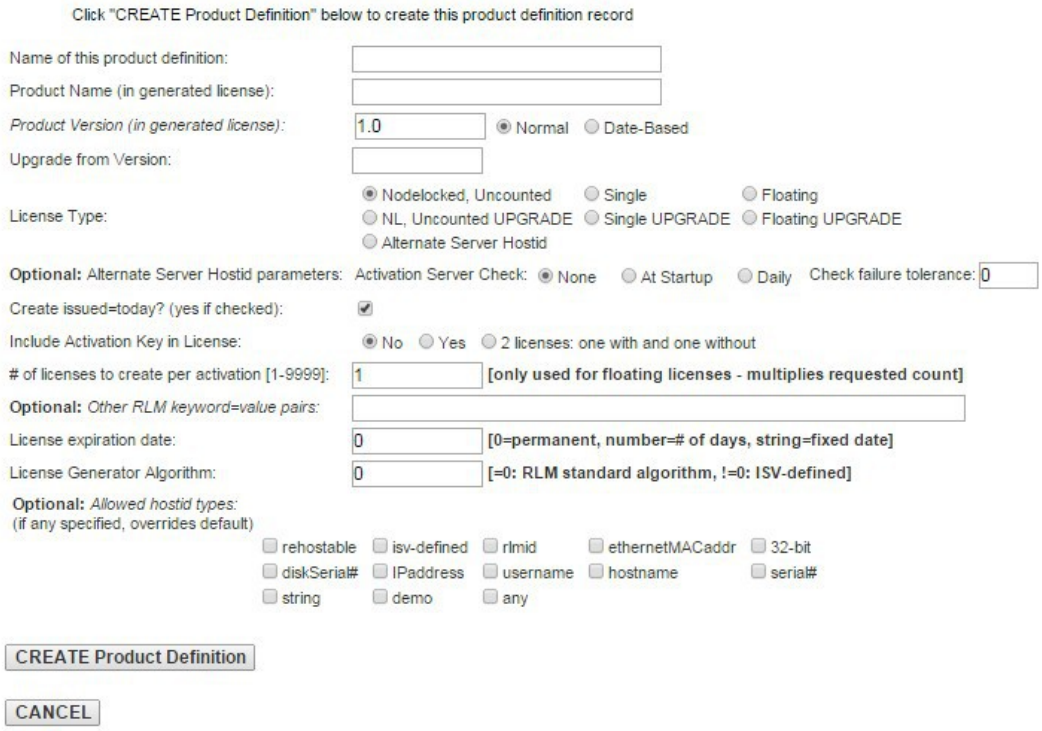
The System Product Definition Defaults form is shown here:



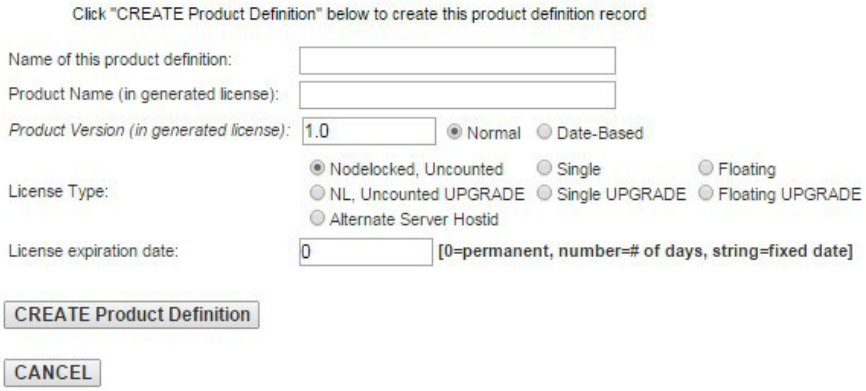
If the “default” checkbox in the left-hand column is checked, then the value(s) in the right-hand column will be used as the default(s) for that field. If the “invisible” checkbox in the center column is checked, then the value(s) in the right-hand field will be used as the default(s) and that field will not appear in the new product or edit product form. (Note: “invisible” sets “default” when saved).

The Activation key default form is similar to the product definition default form. Also, the per-user forms look the same as the system-wide default forms. Note that if a field is set to “invisible” in the system-wide form, that field will not only be invisible in the appropriate form, it also ***will not appear*** on a user's default editing form.

Here is an example (complete) “Create Product” form:



Now, after the “Upgrade From Version”, “Alternate Server Hostid”, “Issued”, “Include Activation Key”, “# licenses per activation”, “Other Parameters”, “License Generator Algorithm”, and “Allowed Hostids” fields were defaulted and made invisible, the form looks like this:



If a field is made invisible after products or activation keys are defined, editing the product or activation key will not affect the prior values of the invisible fields.

The activation key default forms work in exactly the same way as the product definition forms.

Note that the following fields cannot be defaulted:

- product name in product definitions
- license name in product definitions
- active and obsolete flags in product definitions
- activation key name in activation keys
- product name in activation keys
- active flag in activation keys
- # activation keys to create in activation keys
- customer in activation keys

Section 2 – Reference Material

Appendix A – Activation API

The RLM Activation API consists of five calls:

- *rlm_activate()* (the new, preferred activation request call)
- *rlm_act_new_handle()*
- *rlm_act_destroy_handle()*
- *rlm_act_set_handle()*
- *rlm_act_revoke()*

In addition, the following 2 calls are now deprecated:

- *rlm_act_request()* (the old activation request call)
- *rlm_act_refresh()*

These calls are all documented in the RLM Reference Manual.

Appendix B – Example Program

The following is the source code to the activation demo program actdemo: This example program (act_api_example.c) is contained on the RLM kit in the *examples* directory.

```
*****************************************************************************

            COPYRIGHT (c) 2005, 2014 by Reprise Software, Inc.
        This software has been provided pursuant to a License Agreement
        containing restrictions on its use.  This software contains
        valuable trade secrets and proprietary information of
        Reprise Software Inc and is protected by law.  It may not be
        copied or distributed in any form or medium, disclosed to third
        parties, reverse engineered or used in any manner not provided
        for in said License Agreement except with the prior written
        authorization from Reprise Software Inc.

 *****************************************************************************/
/*
 *      Description:    Example client for RLM internet activation
 *
 *      M. Christiano
 *      6/18/07
 *
 *      $Id: act_api_example.c,v 1.21 2013/09/17 01:43:29 matt Exp $
 */

#include "license.h"
#include <stdio.h>
#include <stdlib.h>
#include <string.h>


static int checkstat(RLM_HANDLE, RLM_LICENSE, const char *);
static int doactivation(RLM_HANDLE, const char *);

int
main(int argc, char *argv[])
{
  RLM_HANDLE rh = (RLM_HANDLE) NULL;
  RLM_LICENSE lic = (RLM_LICENSE) NULL;
  int x, stat;
  const char *product = "actdemo";
  int pass;

        for (pass = 1; pass <= 2; pass++)
        {
            if (pass == 2)
            {
/*
 *              We did not get the license on the first pass.  See
 *              if we can activate it now.
 */
                stat = doactivation(rh, product);
                rlm_close(rh);          /* Close the old handle */
                if (stat < 0) exit(2);
            }
/*
 *          NOTE that rlm_init() is in this loop.  This is necessary
 *          because if the license is not acquired on the first pass,
 *          the activation request (above) will create a new
 *          license file.  rlm_init() must be called to see this
 *          new license file.
 */
            rh = rlm_init(".", argv[0], (char *) NULL);
```

```
            stat = rlm_stat(rh);
            if ((pass == 1) && (stat == RLM_EH_READ_NOLICENSE))
            {
/*
 *              We didn't find a license.  Checkout won't work, so just
 *              skip to the 2nd pass where we try to activate it.
 */
                continue;
            }
            else if (stat)
            {
                (void) printf("Error %d initializing license system\n", stat);
                exit(1);
            }

            lic = rlm_checkout(rh, product, "1.0", 1);

            stat = checkstat(rh, lic, product);

            if ((pass == 1) && (stat < 0))
            {
/*
 *              Didn't get the license.  Try a 2nd time to activate it.
 */
                if (lic) rlm_checkin(lic);
                lic = (RLM_LICENSE) NULL;
                continue;
            }
            else if (stat == 0)
            {
/*
 *              We got the license
 */
                break;
            }
        }

        if (stat == 0)
        {
/*
 *              We got the license
 */
                (void) printf("Enter <CR> to continue: ");
                x = fgetc(stdin);
                if (lic) rlm_checkin(lic);
                rlm_close(rh);
        }
        else if (pass == 2)
        {
/*
 *              checkout and/or activation failed.
 */
                (void)
                   printf("Unable to check out/activate \"actdemo\" license\n");
        }
        return(0);
}

static
int
checkstat(RLM_HANDLE rh, RLM_LICENSE lic, const char *name)
{
  int stat;
  char errstring[RLM_ERRSTRING_MAX];

        stat = rlm_license_stat(lic);
        if (stat == 0)
            (void) printf("Checkout of %s license OK.\n", name);
        else
        {
            (void) printf("Error checking out %s license\n", name);
```

```
            (void) printf("%s\n", rlm_errstring(lic, rh, errstring));
        }
        return(stat);
}

#include <time.h>
#ifndef _WIN32
#include <sys/time.h>
#endif

static
int
doactivation(RLM_HANDLE rh, const char *name)
{
  char license[3 * RLM_MAX_LINE + 1];   /* Allow for HOST, ISV, and LICENSE */
  char akey[RLM_MAX_LINE+1];
  int stat = RLM_EH_READ_NOLICENSE;     /* If they say NO, no license */
  int len;

        (void) printf("\nWould you like to activate this license now? ");
        fgets(akey, RLM_MAX_LINE, stdin);
        if (*akey == 'y' || *akey == 'Y')
        {
                (void) printf("Enter Activation key for \"%s\": ", name);
                fgets(akey, RLM_MAX_LINE, stdin);
                len = ((int) strlen(akey)) - 1;
                if (akey[len] == '\n') akey[len] = '\0';
/*
 *              Request the license
 */
                stat = rlm_activate(
                        rh,                             /* RLM handle */
                        "http://www.reprisesoftware.com", /* URL */
                        akey,                           /* Activation key */
                        1,              /* Activation count - # of licenses */
                        license,        /* Space for the returned license */
                        (RLM_ACT_HANDLE) NULL);  /* No other optional params */

                if ((stat == 0) || (stat == 1))
                {
                  char name[100];
                  int attempt;
                  FILE *f;
/*
 *                  Activation was successful.  Write the license out.
 *                  Note in this example, we try the license file name
 *                  aN.lic, and we only try 100 different names.  You
 *                  should change this to whatever naming convention you
 *                  want to use.
 */
                    for (attempt=0; attempt<100; attempt++)
                    {
                        sprintf(name, "a%d.lic", attempt);
                        f = fopen(name, "r");
                        if (f == (FILE *) NULL)
                        {
                          struct tm *t;
#ifdef _WIN32
                          time_t ltime;

                                time(&ltime);
                                t = localtime(&ltime);
#else
                          struct timezone tz;
                          struct timeval tv;
                          time_t x;

                                gettimeofday(&tv, &tz);
                                x = tv.tv_sec;
                                t = localtime((time_t *) &x);
#endif
```

```
                                f = fopen(name, "w");
                                if (f)
                                {
                                    fprintf(f, "This license created by RLM Inte
rnet Activation\n");
                                    if (t)
                                      fprintf(f, "Created on %02d/%02d/%04d at %
02d:%02d\n",
                                                t->tm_mon+1, t->tm_mday,
                                                t->tm_year+1900, t->tm_hour,
                                                t->tm_min);

                                    fprintf(f, "\n%s\n", license);
                                    fclose(f);
                                    printf(
                          "Activation successful, license file \"%s\" written\n",
                                                                    name);
                                    break;
                                }
                                else
                                {
                                    printf(
                                          "Error writing license file \"%s\"\n",
                                                                      name);
                                    stat = -1;
                                    break;
                                }
                            }
                        }
                    }
                    else
                    {
                        printf("\n");
                        switch(stat)
                        {
                          char err[RLM_ERRSTRING_MAX+1];

                            case RLM_EH_CANTCONNECT_URL:
                              printf("You were unable to connect to http://www.repri
sesoftware.com.\n");
                              printf("Please make sure that this system is able to\n
");
                              printf("access the internet and try again.\n");
                              break;


                            case RLM_ACT_NO_KEY_MATCH:
                              printf("The activation key you supplied (%s) was \n",
                                                                        akey);
                              printf("not found.  Please check the key and ensure\n"
);
                              printf("that you have entered it correctly.\n");
                              break;

                            case RLM_ACT_KEY_USED:
                              printf("The activation key you supplied (%s)\n", akey)
;
                              printf("has already been used to activate a license.\n
");
                              printf("Please check the key and ensure that you have
entered it correctly.\n");
                              break;

                            case RLM_EH_BAD_HTTP:
                              printf("Bad HTTP transaction\n%s\n",
                                                rlm_errstring(0, rh, err));
                              break;

                            default:
                              printf("Error %d requesting activation\n%s\n",
```

```
                                        stat,
                                        RLM_ACT_ERR(stat) ?
                                            rlm_act_errstring(stat) :
                                            rlm_errstring(0, rh, err));
                        break;

                    }
                    printf("\n");
                }
        }
        return(stat);
}
```

Appendix C – License Rehosting

Rehostable Licenses

Activation can be set up to support rehosting of licenses by the end user without ISV involvement. This involves the use of the rehostable hostid type, which is new in RLM v9.3. A rehostable hostid is requested at the time that *rlm_activate()* is called, and the activation server issues the license to this hostid. Later, if the user wishes to move this license to another system, the *rlm_act_revoke()* call can be used to revoke the rehostable license and allow it to be activated on another system. Note that the rehostable hostid is a copy-protected file, and if the system should crash, the hostid cannot be retrieved. As an ISV, you will need a policy to deal with this situation – e.g., to give the customer another activation in order to continue. Since this situation should be rare, a non-automated request procedure might be quite sufficient.

Rehostable license support is limited to nodelocked, uncounted (or single) licenses and product definitions that specify only a single license. In addition, only one version of any product can exist with a rehostable hostid on a system at any given time. Attempting to activate a 2nd product with a different version number will give an RLM_EH_REHOST_EXISTS (-153) error from activation. Attempting to activate either a counted license or a product with multiple licenses will give an RLM_ACT_NO_REHOST (-1035) error. Finally, you should always use a fixed expiration date if you are using the **Normal** fulfillment type for Rehostable licenses. If you want an expiration date that is relative to the time of fulfillment, then you should use a **ReActivate** fulfillment type and (beginning in RLM v11.0) set the count to the # of times you want your customer to be able to revoke and re-activate the license.

Note that you have a choice when it comes to the revocation of rehostable licenses. By default, RLM Activation Pro will delete the hostid when *rlm_act_revoke()* is called, but if the license has expired, the Activation Pro server will not return count to the activation key, and it will return the status RLM_ACT_REVOKE_TOOLATE to the application. However, you can configure Activation Pro to allow the server to return count to the activation key in this case, and return a good status to the application. See the Database section of the Your Activation Database chapter on page 40 for more information on setting this parameter.

How to activate with a rehostable hostid

In order to create a rehostable hostid, you must set the activation handle to indicate that you want to activate with a rehostable hostid. To do this, you use code similar to the following (this code is from the example in the section below):

```
act_handle = rlm_act_new_handle(rh);
rlm_act_set_handle(act_handle, RLM_ACT_HANDLE_REHOST,
                                                (void *) 1);
stat = rlm_activate(
        rh,             /* RLM handle */
        "http://www.reprisesoftware.com", /* URL */
        akey,           /* Activation key */
        1,              /* count - # of licenses */
        license,        /* Space for the returned license */
        act_handle      /* Activation handle data */
        );

rlm_act_destroy_handle(act_handle);   /* Done with this */
```

If you do not set the handle to RLM_ACT_HANDLE_REHOST before the rlm_activate() call, you will not create a rehostable hostid and the license will be activated using the default hostid on the machine.

What to do if you don't want to use rehostable hostids all the time

Since a rehostable hostid must be requested at activation time by setting the handle before you call *rlm_activate()*, you must know before activation time whether you want to use a normal or a rehostable hostid. If you know this on a product-by-product basis, you can set the allowed hostids in the product definition, setting only "rehostable" in the product definition for a product that always uses rehostable hostids. If your products can use rehostable or regular hostids, but you know which at activation key creation time, you can set the allowed hostids in the activation key. In either case, use an algorithm similar to the following:

1. attempt the activation for a "normal" hostid. If this succeeds, you are done.
2. If the normal hostid activation fails, re-attempt the activation specifying a rehostable hostid.
3. If both attempts (1) and (2) fail, then the product definition doesn't specify a rehostable hostid and the default hostid on this system isn't in the list of allowed hostids, either.

If you don't know before activation time, you will have to prompt the user and allow both rehostable and regular hostids in the product definition/activation key, then make the *rlm_activate()* request with the correct parameter.

How do I set up activation pro to accept the correct type of hostid?

This is really straightforward.

- the only way to get a rehostable hostid is to request it explicitly, in which case, only the rehostable hostid will get sent to the actpro server.
- if you don't request it, RLM will fill in the list of "standard" hostids and send those to the server

Now, as far as your activation pro server is concerned:

- if your activation key specifies rehostables only, then only a rehostable will work.
- if your activation key specifies other types of hostids,, then only those hostids will work
- if your activation keys specifies nothing, then the hostid requirement falls back to the product definition:
  - if your product definition specifies rehostables only, then only a rehostable will work.
  - if your product definition specifies other types of hostids,, then only those hostids will work
- if neither the activation key nor the product definition specifies allowed hostids, then the default is used, which is set to include rehostables (unless you change it in rlm_isv_config.c)

What happens when my customer inadvertently deletes the rehostable hostid data?

When you create a rehostable license, you write a license file which specifies a rehostable hostid. The rehostable hostid is a copy-protected directory hierarchy on the computer. Sometimes, your customer will delete the license file or delete/invalidate the hostid. What do you do in this case?

If your customer deletes the license file, you first have to determine the hostid that was used for this product, then you must contact the activation server to retrieve the license. You can do this by first retrieving the rehostable hostid with the new *rlm_get_rehost()* API call (new in v11.1), then asking the activation server for the license activated on that hostid with the new *rlm_act_keyvalid_license()* call. You will need to know the product name and activation key to do this. The following code snippet illustrates this operation:

```
char product_name[RLM_MAX_PRODUCT+1];
char hostid[RLM_MAX_HOSTID_STRING+1];
char license[RLM_ACT_MAX_LICENSE+1];
char *akey;
stat = rlm_get_rehost(rh, product_name, hostid);
akey = "activation key";
if (!stat)
{
        /* Request the license from the activation server */
        stat = rlm_act_keyvalid_license(rh, "www.yourURL.com", akey, hostid,
                                                                license);
        if (!stat)  /* Write license out to license file */
}
```

If your customer deletes or invalidates the hostid, you have the option of allowing them to revoke the license **ON THE SAME MACHINE** where they activated it. This capability is new in RLM v11.1, and requires both an 11.1 client and and 11.1 activation server. You do this with the new *rlm_act_revoke_reference()* call. Note that you should always attempt to do the *rlm_act_revoke()* operation first; only if it returns RLM_EH_CANT_GET_REHOST should you call *rlm_act_revoke_reference()*. The following sample code snippet illustrates this operation:

```
stat = rlm_act_revoke(rh, "www.yourURL.com", product_name);
if (stat == RLM_EH_CANT_GET_REHOST)
{
        /* Some error message here, perhaps */
        stat = rlm_act_revoke_reference(rh, "www.yourURL.com", product_name);
}
```

Rehostable License Example code

An example for rehostable licenses is on the RLM kit in the examples directory, called rehost_example.c It is repeated here. The important calls to do the rehosting are highlighted in yellow:

```
/*****************************************************************************

        COPYRIGHT (c) 2005, 2014 by Reprise Software, Inc.
    This software has been provided pursuant to a License Agreement
    containing restrictions on its use.  This software contains
    valuable trade secrets and proprietary information of
    Reprise Software Inc and is protected by law.  It may not be
    copied or distributed in any form or medium, disclosed to third
    parties, reverse engineered or used in any manner not provided
    for in said License Agreement except with the prior written
    authorization from Reprise Software Inc.

 *****************************************************************************/
/*
```

```
 *      Description:   Example client for RLM license rehosting/revoking
 *
 *      M. Christiano
 *      11/18/11 - modified from the regular activation example
 *
 *      $Id: rehost_example.c,v 1.3 2012/01/04 00:44:04 matt Exp $
 */

#include "license.h"
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

static int checkstat(RLM_HANDLE, RLM_LICENSE, const char *);
static int doactivation(RLM_HANDLE, const char *);
static int dodeactivation(RLM_HANDLE, const char *);

int
main(int argc, char *argv[])
{
  RLM_HANDLE rh = (RLM_HANDLE) NULL;
  RLM_LICENSE lic = (RLM_LICENSE) NULL;
  int stat;
  const char *product = "actdemo";
  int pass;

        for (pass = 1; pass <= 2; pass++)
        {
            if (pass == 2)
            {
/*
 *              We did not get the license on the first pass.  See
 *              if we can activate it now.
 */
                stat = doactivation(rh, product);
                 rlm_close(rh);        /* Close the old handle */
                if (stat < 0) exit(2);
            }
/*
 *          NOTE that rlm_init() is in this loop.  This is necessary
 *          because if the license is not acquired on the first pass,
 *          the activation request (above) will create a new
 *          license file.  rlm_init() must be called to see this
 *          new license file.
 */
            rh = rlm_init(".", argv[0], (char *) NULL);
            stat = rlm_stat(rh);
            if ((pass == 1) && (stat == RLM_EH_READ_NOLICENSE))
            {
/*
 *              We didn't find a license.  Checkout won't work, so just
 *              skip to the 2nd pass where we try to activate it.
 */
                continue;
            }
            else if (stat)
            {
                (void) printf("Error %d initializing license system\n", stat);
                exit(1);
            }

            lic = rlm_checkout(rh, product, "1.0", 1);

            stat = checkstat(rh, lic, product);

            if ((pass == 1) && (stat < 0))
            {
/*
 *              Didn't get the license.  Try a 2nd time to activate it.
 */
                 if (lic) rlm_checkin(lic);
```

```
                lic = (RLM_LICENSE) NULL;
                continue;
            }
            else if (stat == 0)
            {
/*
 *              We got the license
 */
                break;
            }
        }

        if (stat == 0)
        {
/*
 *              We got the license
 */
                if (lic) rlm_checkin(lic);
                printf("License valid.\n");
                stat = dodeactivation(rh, product);
                rlm_close(rh);
        }
        else if (pass == 2)
        {
/*
 *              checkout and/or activation failed.
 */
                (void)
                   printf("Unable to check out/activate \"actdemo\" license\n");
        }
        return(0);
}

static
int
checkstat(RLM_HANDLE rh, RLM_LICENSE lic, const char *name)
{
  int stat;
  char errstring[RLM_ERRSTRING_MAX];

        stat = rlm_license_stat(lic);
        if (stat == 0)
            (void) printf("Checkout of %s license OK.\n", name);
        else
        {
            (void) printf("Error checking out %s license\n", name);
            (void) printf("%s\n", rlm_errstring(lic, rh, errstring));
        }
        return(stat);
}

#include <time.h>
#ifndef _WIN32
#include <sys/time.h>
#endif
int
doactivation(RLM_HANDLE rh, const char *name)
{
  char license[3 * RLM_MAX_LINE + 1]; /* Allow for HOST, ISV, and LICENSE */
  char akey[RLM_MAX_LINE+1];
  int stat = RLM_EH_READ_NOLICENSE;   /* If they say NO, no license */
  int len;
  RLM_ACT_HANDLE act_handle;

        (void) printf("\nWould you like to activate this license now? ");
        fgets(akey, RLM_MAX_LINE, stdin);
        if (*akey == 'y' || *akey == 'Y')
        {
                (void) printf("Enter Activation key for \"%s\": ", name);
                fgets(akey, RLM_MAX_LINE, stdin);
                len = ((int) strlen(akey)) - 1;
```

```
                if (akey[len] == '\n') akey[len] = '\0';
/*
 *              Request the license.  First make a handle, and tell it we
 *              want a rehostable hostid to activate.
 */
                act_handle = rlm_act_new_handle(rh);
                rlm_act_set_handle(act_handle, RLM_ACT_HANDLE_REHOST,
                                                        (void *) 1);
/*
 *               Note - you would normally never need to make this next call,
 *               but we do it here so that we can connect to the license
 *               generator on the reprise demo site.
 */
                rlm_act_set_handle(act_handle, RLM_ACT_HANDLE_ISV,
                                                (void *) "rlmactdemo");

                stat = rlm_activate(
                        rh,             /* RLM handle */
                        "http://www.reprisesoftware.com", /* URL */
                        akey,           /* Activation key */
                        1,              /* count - # of licenses */
                        license,        /* Space for the returned license */
                        act_handle      /* Activation handle data */
                        );

                rlm_act_destroy_handle(act_handle);   /* Done with this */

                 if ((stat == 0) || (stat == 1))
                 {
                   char name[100];
                   int try;
                   FILE *f;
/*
 *                   Activation was successful.  Write the license out.
 *                   Note in this example, we try the license file name
 *                   aN.lic, and we only try 100 different names.  You
 *                   should change this to whatever naming convention you
 *                   want to use.
 */
                      for (try=0; try<100; try++)
                      {
                          sprintf(name, "a%d.lic", try);
                          f = fopen(name, "r");
                          if (f == (FILE *) NULL)
                          {
                            struct tm *t;
#ifdef _WIN32
                            time_t ltime;

                                  time(&ltime);
                                  t = localtime(&ltime);
#else
                            struct timezone tz;
                            struct timeval tv;
                            time_t x;

                                  gettimeofday(&tv, &tz);
                                  x = tv.tv_sec;
                                  t = localtime((time_t *) &x);
#endif

                                  f = fopen(name, "w");
                                  if (f)
                                  {
                                      fprintf(f, "This license created by RLM Internet
Activation\n");
                                      if (t)
                                        fprintf(f, "Created on %02d/%02d/%04d at %02d:
%02d\n",
                                                t->tm_mon+1, t->tm_mday,
                                                t->tm_year+1900, t->tm_hour,
```

```
                                        t->tm_min);

                                fprintf(f, "\n%s\n", license);
                                fclose(f);
                                printf(
                        "Activation successful, license file \"%s\" written\n",
                                                        name);
                                break;
                            }
                            else
                            {
                                printf(
                                      "Error writing license file \"%s\"\n",
                                                                name);
                                stat = -1;
                                break;
                            }
                        }
                    }
                }
                else
                {
                    printf("\n");
                    switch(stat)
                    {
                      char err[RLM_ERRSTRING_MAX+1];

                        case RLM_EH_CANTCONNECT_URL:
                          printf("You were unable to connect to
http://www.reprisesoftware.com.\n");
                          printf("Please make sure that this system is able to\n");
                          printf("access the internet and try again.\n");
                          break;

                        case RLM_ACT_NO_KEY_MATCH:
                          printf("The activation key you supplied (%s) was \n",
                                                                akey);
                          printf("not found.  Please check the key and ensure\n");
                          printf("that you have entered it correctly.\n");
                          break;

                        case RLM_ACT_KEY_USED:
                          printf("The activation key you supplied (%s)\n", akey);
                          printf("has already been used to activate a license.\n");
                          printf("Please check the key and ensure that you have entered it
correctly.\n");
                          break;

                        case RLM_EH_BAD_HTTP:
                          printf("Bad HTTP transaction\n%s\n",
                                            rlm_errstring(0, rh, err));
                          break;

                        default:
                          printf("Error %d requesting activation\n%s\n",
                                          stat,
                                          RLM_ACT_ERR(stat) ?
                                              rlm_act_errstring(stat) :
                                              rlm_errstring(0, rh, err));
                          break;

                    }
                    printf("\n");
                }
        }
        return(stat);
}

int
dodeactivation(RLM_HANDLE rh, const char *name)
{
```

```
  int stat = RLM_EH_READ_NOLICENSE;   /* If they say NO, no license */
  char x[100];

        (void) printf("\nWould you like to deactivate the \"%s\" license now? ",
                                                                    name);
        fgets(x, RLM_MAX_LINE, stdin);
        if (*x == 'y' || *x == 'Y')
        {
/*
 *              Request the deactivation.
 */
                stat = rlm_act_revoke(rh, "www.reprisesoftware.com",
                                                          (char *) name);

                if (stat == 0)
                {
/*
 *                      revoke (Deactivation) was successful.
 */
                        printf("License successfully revoked\n");
                }
                else
                {
                  char err[RLM_ERRSTRING_MAX+1];

                        printf("\n");
                        printf("Error %d requesting license revoke\n%s\n",
                                        stat,
                                        RLM_ACT_ERR(stat) ?
                                            rlm_act_errstring(stat) :
                                            rlm_errstring(0, rh, err));
                        printf("\n");
                }
        }
        return(stat);
}
```

Rehostable hostids on disconnected systems

Sometimes it is desirable to create a rehostable hostid on a system with no access to the internet. RLM allows you to do this; it is a 3-step process, with the first step performed on the disconnected system, then the next on an internet-connected system. Finally, the resulting license must be transferred back to the disconnected system. Similarly, revoking a rehostable hostid on a disconnected system is a 2-step process – first the revoke is started on the disconnected system, then finished on a system with a connection to the internet.

Creating a rehostable hostid on a disconnected system.

Normally, a rehostable hostid is activated by calling rlm_activate() with a handle that specifies the RLM_ACT_HANDLE_REHOST parameter and the activation key. When the system is disconnected from the internet, the procedure is as follows:

Step 1: call rlm_activate() with a handle that specifies RLM_ACT_HANDLE_REHOST, RLM_ACT_HANDLE_DISCONN, and the product name in RLM_ACT_HANDLE_PRODUCT. You must specify the product name on the disconnected system. The activation key is not required. rlm_activate() will return a set of data (in the "license" parameter) which you must then transmit to the system that is connected to the internet.

Step 2: On the internet-connected system, call rlm_activate() with a handle that specifies RLM_ACT_HANDLE_REHOST, RLM_ACT_HANDLE_DISCONN, nothing in the product

name, and the data from step 1 in RLM_ACT_HANDLE_HOSTID_LIST. The activation key is required in this call, and it must correspond to the product name specified in step 1. This call will return the license (in the "license" parameter).

Step 3: take the license from step 2, and install it on the disconnected system.

For example:

(run this on the disconnected system)

```
#define URL "your-activation-server-URL"
#include "license.h"
RLM_ACT_HANDLE act_handle;
char data[3*(RLM_MAX_LINE+1)];
int stat;

    rh = rlm_init();
    act_handle = rlm_act_new_handle(rh);
    stat = rlm_act_set_handle(act_handle, RLM_ACT_HANDLE_REHOST, 1);
    stat = rlm_act_set_handle(act_handle, RLM_ACT_HANDLE_DISCONN, 1);
    stat = rlm_act_set_handle(act_handle, RLM_ACT_HANDLE_PRODUCT, "product-name");
    stat = rlm_activate(rh, URL, "", 0, data, act_handle);
    (void) rlm_act_destroy_handle(act_handle);
```

(Take the result in "data" and run this on the internet-connected system)

```
#define URL "your-activation-server-URL"
#include "license.h"
RLM_ACT_HANDLE act_handle;
char license[3*(RLM_MAX_LINE+1)];
char activation_key;  /* should be set to the activation key */
int stat;

    rh = rlm_init();
    act_handle = rlm_act_new_handle(rh);
    stat = rlm_act_set_handle(act_handle, RLM_ACT_HANDLE_REHOST, 1);
    stat = rlm_act_set_handle(act_handle, RLM_ACT_HANDLE_DISCONN, 1);
    stat = rlm_act_set_handle(act_handle, RLM_ACT_HANDLE_HOSTID_LIST, data);
    stat = rlm_activate(rh, URL, activation_key, 0, license, act_handle);
    (void) rlm_act_destroy_handle(act_handle);
```

Finally, take the result in "license" and install it in a license file on the disconnected system.

Revoking a rehostable hostid on a disconnected system.

Normally, a rehostable hostid is revoked by calling rlm_act_revoke() or rlm_act_revoke_reference() with the server's URL and the product name. When the system is disconnected from the internet, the procedure is as follows:

Step 1: on the disconnected system, call rlm_act_revoke_disconn(), specifying a NULL URL and the name of the product in the third parameter. This will return a verification string in the last parameter.

Step 2: transfer the verification string to the the internet-connected system, call rlm_act_revoke_disconn() specifying the URL and the verification string in the third parameter.

For example:

(run this on the disconnected system)

```
#include "license.h"
char verification[3*(RLM_MAX_LINE+1)];
int stat;

    rh = rlm_init();
    stat = rlm_act_revoke_disconn(rh, "", "product-name", verification);
```

(Take the result in "verification" and run this on the internet-connected system)

```
#define URL "your-activation-server-URL"
#include "license.h"
char license[3*(RLM_MAX_LINE+1)];
int stat;

    rh = rlm_init();
    stat = rlm_act_revoke_disconn(rh, URL, verification, (char *) NULL);
```

Appendix D – Activation Database

RLM activation uses a MySQL database to store all data. There are a total of 19 database tables:

- active_users - the currently active users
- audit – audit trail
- badhost – used to automatically blacklist IP addresses
- blacklist - blacklist definitions,
- company – table of customer companies
- contact – table of customer contacts
- contact_types – table of contact types
- debuglog – license generator debug logging information
- defaults – default values for product definition and activation keys
- keyd - The Activation Key definitions,
- keyf - The Activation key fulfillment status,
- licf - License Fulfillment data
- prod - The product definitions,
- report_cols – database columns which appear in reports
- report_select – selection criteria for reports
- report_types – report type definitions
- setup - Activation setup data,
- stats – license generator statistics,
- users - Table of authorized users,

Each table indicates whether the admin tool and/or the license generator read and write to it.

The contents of these tables is as follows:

active_users - active **rlc** users

This table is used only by the admin tool. Unused by the license generator.
Contents:

column	meaning	data type
user	username logged in	string
timestamp	Time user logged in	integer

audit – database Audit Trail (v11.2)

This table is used only by the admin tool. Unused by the license generator.
Contents:

column	meaning	data type
date	Time of entry	integer
who	username logged in	string
what	Operation performed	string

tablekey	Key in table modified	string
tablename	Table modified	string

badhost – automatic blacklisting data

This table is used by both the admin tool. and the license generator.
Contents:

column	meaning	data type
ip	IP address	string
first	First transaction	integer
last	Last transaction	integer
count	Transaction count	integer

blacklist - blacklist definitions, one row per blacklisted host.

This table is written only by the admin tool. Read-only by the license generator.
Contents:

column	meaning	data type
id	row id	integer
domain	the hostname/domain of the blacklisted host. If this string matches any part of the requesting host, that host is not allowed to use the activation server	string (255). No spaces.
type	Type of blacklist entry	integer

company – customer company information

This table is used only by the admin tool. Unused by the license generator.
Contents:

column	meaning	data type
company_id	Company ID (primary key)	integer
company	Company name	string (60)
addr1	Address line 1	string (60)
addr2	Address line 2	string (60)
addr3	Address line 3	string (60)
city	City	string (60)
state	State	string (60)
zip	Zip/Postal code	string (60)

country	Country	string (60)
c_notes	Misc notes.	string (255)
u1	User-defined.	string (60)
u2	User-defined.	string (60)
u3	User-defined.	string (60)
u4	User-defined.	string (60)
u5	User-defined.	string (60)
u6	User-defined.	string (60)
u7	User-defined.	string (60)
u8	User-defined.	string (60)
u9	User-defined.	string (60)
u10	User-defined.	string (60)

contact – customer contact person information

This table is used only by the admin tool. Unused by the license generator.
Contents:

column	meaning	data type
contact_id	Contact ID (primary key)	integer
contact	Contact person name	string (60)
contact_type	Contact type	string (60)
title	Job Title	string (60)
phone	Phone #	string (20)
fax	Fax #	string (20)
email	Email address	string (60)
notes	General notes	string (60)
company_id	ID of associated company record	integer

contact_types – customer contact type definitions

This table is used only by the admin tool. Unused by the license generator.
Contents:

column	meaning	data type
contact_type	Contact type	string (60)

debuglog – license generator debug log

This table is written by the license generator and admin tool. Read by the admin tool only.
Contents:

column	meaning	data type
date	Date/time of entry	integer
logdata	The debug data logged	string (500)

defaults – default specifications for products and activation keys

This table is read and written only by the admin tool.
Contents:

column	meaning	data type
id	Row unique id	integer
user	Whose default (#SYS# for system default)	string (30)
tab	Which table (prod or keyd)	string(30)
col	Column in table	string (30)
value	Default value	string(80)
invis	Field is invisible	int(1)

keyd - activation key definitions, one row per activation key.

This table is written only by the admin tool. Read-only by the license generator.
Contents:

column	meaning	data type
akey	the activation key (35 character maximum)	string
active	0 if this definition is inactive, 1 if active	integer
product id	product identifier, from product definitions	integer
count	the # of activations allowed, 0 = unlimited. Limited to 1 for Refresh-type activations.	integer
type	0 - Normal, 1 - Reactivate, 2 = Refresh	integer
rehosts	For revoked hostids, the number of hostid changes allowed	integer
lastdate	Expiration date of activation key - key cannot be used after this date	string
exp	If a number, number of days to expiration of the license after product activation. If == 0, the license does not expire. If a standard RLM date format, it is a fixed expiration date. If specified, this expiration date overrides	string

	the expiration date in the product definition.	
kver	License version, to override product definition	string(11)
kver_type	License version type	integer
white	Domain whitelist for key - only domain names appearing in this list can use the key. This string can be a space-separated list. If any component of the string matches the requesting hostname, that host is allowed to activate using this key.	string (100)
misc	any optional RLM license attributes	string (255)
notes	optional notes to allow you to identify the key - unused by RLM	string (100)
contact_id	Contact ID of owner of activation key	integer
key_allowed_hostids	Allowed Hostids for this activation key (overrides product definition)	integer
Kcreate	Key creation time	char(11)
ash_sn	Serial number for Alternate Server hostid	integer
user_def	User-defined field (unused by Activation Pro)	char(11)

keyf - activation key fulfillment information, one row per activation key.

This table is written by both the license generator and rlc.
Contents:

column	meaning	data type
akey	the activation key	string
num	Number of fulfillments from this key	integer
date	For Reactvation: original expiration date of the license. For Refresh: last date license refreshed	string
lasthostid	For Refresh type: Last hostid fulfilled	string (80)
num_revoked	Number of licenses revoked	integer

licf - license fulfillment information, one row per activation.

This table is written by both the license generator and rlc.
Contents:

column	meaning	data type
id	row ID	integer
akey	the activation key	string
product_id	product identifier	integer
count	number of licenses activated	integer

time	time() value when activation was done	time_t (integer)
license_hostid	hostid for license	string
reference_hostid	Reference hostid for rehostable licenses	string
expdate	Expiration date of first license, in fmt yyyy-mm-dd	string
hostname	hostname on activated machine	string
remote_host	IP address (typically) of machine requesting activation	string
log	Log info from rlm_act_request() call	string
license	generated license. For floating licenses, this is just the LICENSE line	string
last_fulfill	time() value at last fulfillment. (v9.3)	time_t (integer)
num_fulfills	Number of fulfillments of this license (v9.3)	integer
revoked	“license was revoked” flag (1=revoked) (v9.3)	integer
revoked_time	Time license was revoked	integer
last_check	Last time an rlm_act_keyvalid() called on this fulfillment	integer

prod - product definitions, one row per product definition.

This table is written only by the admin tool. Read-only by the license generator.
Contents:

column	meaning	data type
id	activation product identifier	integer
active	0 if this definition is inactive, 1 if active (product definitions are not deleted). The license generator doesn't create licenses for inactive products	integer
obsolete	1 if this product definition is obsolete, meaning it can't be used to create new activation keys	integer(1)
name	Product Definition name	string
version	the RLM product version	string
version_type	0 – normal version, non-zero – date-based version	integer
upgrade_version	Upgrade-from version number (UPGRADE licenses only)	string
exp	If a number, number of days to expiration of the license after product activation. If == 0, the license does not expire. If a standard RLM date format, it is a fixed expiration date.	string
lictype	0 = floating, 1 = node-locked, uncounted, 2 = node-locked, counted (unimplemented), 3 = single, 4 = UPGRADE (floating)	integer
issued	0 - do not include "issued=" in license, 1 - include "issued=[today's date]" in license	integer
add_akey	0 – do not include “akey=” in license, 1 – include “akey=”	integer
product	License product name	string
misc	any optional RLM license attributes	string
generator	Used to specify alternate license generators. 0=standard RLM generator	integer
prod_id	id of the Primary License of this product definition. 0 for the primary license.	integer
nlic	Number of licenses to create per fulfillment request. Multiplied by request count. Unused for nodelocked or single licenses.	integer
prod_allowed_hostids	Bitmap of allowed hostid types. If 0, all hostid types are allowed	Integer
pcreate	Record creation date	String
ash_type	Alternate server hostid type	integer
ash_tolerance	Alternate server hostid tolerance	integer

report_cols– database columns which appear in report.

This table is read-only by rlc. Unused by the license generator.
Contents:

column	meaning	data type
id	Primary key	int
report	Report name	string (40)
displayorder	Order to display selections	int (2)
tablecolumn	Column name from table	string (40)
display	Report column header text	string (30)
var	POST variable name	string (20)
is_int	If non-zero, column contains integer data	Int (1)

report_select – report selection definitions.

This table is read-only by rlc. Unused by the license generator.
Contents:

column	meaning	data type
id	Primary key	int
report	Report name	string (40)
fixed	Is selection criteria fixed or variable	int (1)
displayorder	Order to display selections	int (2)
display	Text to display in UI	string (40)
var	POST variable name	string (20)
size	Size of text box in UI	int (3)
op	Table for report data	string (20)
value	Value for fixed selection criteria	string (64)

report_types – report type definitions.

This table is read-only by rlc. Unused by the license generator.
Contents:

column	meaning	data type
report	Report name	string (40)
tablename	Table for report data	string (40)

displayorder	Order to display report names	int (2)
report_header	Display header for report	string (80)

setup - RLC setup definitions, one row per setup item.

This table is written only by the admin tool. Read-only by the license generator.
This table contains data such as the activation URL for refresh, Activation Pro version, etc.
Contents:

column	meaning	data type
what	Description of the data item	string (30)
data	The actual data	string (64)

stats – Activation statistics.

This table contains license generation statistics. This table is read-only by rlc. Read-write by the license generator.
Contents:

column	meaning	data type
date	Date in the format" yyyy-mm-dd hh". This record collects statistics for 60 minutes starting at the hour indicated. Primary key	char(13)
total	Total transactions with the activation server	int (11)
good	Completed license fulfillments	int (11)
badkey	Unknown activation key supplied	int (11)
blacklist	# of hosts automatically blacklisted	integer
checks	Reprise pings the hosted activation servers with an invalid activation key to check that the website is up, the license generator is working and the database is working.. This is a bad key where the key supplied is "rsi-check-act-server"	int (11)
numfulfill	Number of "num fulfill" commands executed. Used by rlm_act_revoke()	int (11)
rmfulfill	Number of "remove fulfill" commands executed. Used by rlm_act_revoke()	int (11)
prod	Number of "list product" commands executed. Used by rehostable hostid activation.	int (11)
badtime	Number of otherwise good requests from clients with clocks which are off by more than 7 days	int (11)

info	# of getinfo requests	integer
keyvalid	# of keyvalid requests	integer

users - authorized **rlc** users

This table is used only by the admin tool. Unused by the license generator.
Contents:

column	meaning	data type
user	username	string
password	user's encrypted password	string
userid	User Identifier	string
userlevel	User access (1=view, 2=edit, 9=admin)	integer
email	User's email address	string(50)
timestamp	Last time user logged in	integer
contact_id	User's contact_id	integer
vistabs	Visible Tabs bitmask	integer

Appendix E – Upgrading from RLM Internet Activation

If you are a current RLM Internet Activation customer, you will have a production activation database. This section describes how to upgrade that database to *RLM Activation Pro*.

The upgrade procedure should be performed ***after*** you install *RLM Activation Pro,* but ***before*** you add any data to the database.

To upgrade your production database, change directory to the directory containing your RLM Internet Activation data files, and perform the following steps:

- cd ***activation*** **directory**
- run rlmact2sql from the activation pro directory:

```
% activation_pro_dir/rlmact2sql isvname
```

(where *isvname* is your ISV name)

This will create a database update file named *isvname*_db.sql

- Login to activation pro as the admin user.
- In the login section of activation pro, select "Administer Database"
- In the section of the next screen, under "Upload Activation database data", click to browse to the *isvname*_db.sql file you just created.
- Select "Upload File" to the right-hand side of the filename.

That is it. rlmact2sql will read your existing data from the import file and create the data in your new Activation Pro MySQL database.

Appendix F - Frequently-Asked Questions

Reprise Software maintains a list of frequently-asked questions on our website. For the current list of Frequently-Asked Questions, please see our website.

For ISVs, see the FAQ at our website at:

http://www.reprisesoftware.com/publisher/license-management-faq.php

For Your Customers, see the License Administrator FAQ at

http://www.reprisesoftware.com/admin/software-licensing-faq.php

Appendix G – Document Revision History

v11.3 – Apr-2015 – v11.3 release (BL1)

v11.2 – Nov-2014 – v11.2 release (BL2)

v11.2 – Aug-2014 – v11.2 beta release (BL1)

v11.1 – June-2014 – v11.1 release (BL2)

v11.1 – Apr-2014 – v11.1 beta release (BL1)

v11.0 – 7-Feb-2014 – v11.0 release (BL2)

v11.0 – Dec-2013 – v11.0 beta release (BL1)

v10.0 – 16-Jan-2013 – v10.0 release (BL2)

v10.0 – 19-Nov-2012 – v10.0 beta release (BL1)

V9.4– 28-Sept-2012 - v9.4 release (BL2)

V9.3– 15-February-2012 - v9.3 release (BL2)

v9.3 – 21-November-2011 – First v9.3 beta release (BL1).

V9.2 – 28-September-2011 - v9.2 release (BL2)

v9.2 - September-2011 - First v9.2 release (BL1) – first release with PDF manual.

V9.1 – 2-May-2011 - v9.1 release (BL3) (HTML wiki manual)

v9.1 - Mar-2011 - Second v9.1 beta release (BL2) – second beta release (HTML wiki manual)
v9.1 - Feb-2011 - First v9.1 beta release, first release with Activation PRO (BL1) (HTML wiki manual)

Appendix H – Diagnosing Activation Problems

Beginning in RLM v11.0, the activation server writes debugging information to the database. This information contains, at a minimum, some request data and the status response. To see this debug data, select the "Admin" tab, then select the "Debugging" tab in the 2nd line. Here you will see 3 buttons: "View Debug Log", "Enable Debug Logging", and "Disable Debug Logging". If you want to see the debug output, press the "Enable Debug Logging", then "View Debug Log". You can clear the log from within the page where you are viewing it. Also, you should not run continuously with debug logging turned on, since this will tend to fill your database with log data.

On the client side, you can set the environment variable "RLM_ACT_DEBUG" before running your activation utilty (or rlm). With this environment variable set, debug output will be written to the process's stdout.

When you call rlm_act_request(), errors can come from either the server side, or locally. This section lists most of these errors and what they mean.

Errors detected and returned by the activation server (ISV_mklic):

Status return	Problem	Solution
RLM_EH_BADPARAM (-123)	Error creating license	If RLM_EH_BADPARAM is returned without a corresponding secondary error, this is an error on the server side generating the license. Most likely, this means that some part of the license contains illegal characters – generally in the "extra" parameters supplied in the product definition, activation key, or in the rlm_act_request() call itself.
RLM_EH_ACT_BADLICKEY (-135)	Bad license key in activation binary/settings file.	Your *ISV_mklic* binary (or generator settings file) has incorrect RLM license keys. Re-build and install the new binary or settings file.
RLM_EH_BAD_HTTP (-136)	An unknown HTTP error was returned to the RLM activation client software.	This is usually an installation problem. Some possibilities are: • you have specified an ISV name which is incorrect • the ISV web server is not configured correctly • the *ISV_mklic* binary is not for the correct architecture
RLM_EH_ACT_UNLICENSED (-159)	Activation platform unlicensed for RLM (in license_to_run.h)	This happens if you have built a custom license generator (for example, to support an ISV-defined hostid) and you are running actpro on a platform which isn't licensed for RLM.

Status return	Problem	Solution
RLM_EH_ACTPRO_UNLICENSED (-160)	Activation Pro license missing or invalid	Check the actpro.lic file in the cgi-bin directory. Make sure it is for "rlm_actpro_ISVNAME" where ISVNAME is your ISV name, and that it hasn't expired.
RLM_ACT_NO_KEY (-1002)	No activation key supplied to the activation server.	This is an error in the configuration of your software
RLM_ACT_NO_PROD (-1003)	Activation key specifies a non-existent product.	Make sure your activation key refers to an existing product. This is a setup problem.
RLM_ACT_CANT_WRITE_KEYS (-1004)	Cannot write the license key fulfillment data.	This is an internal server error. Check to make sure the MySQL server is running correctly.
RLM_ACT_KEY_USED (-1005)	Activation key already used	The supplied activation key has already been used and no more activations are available.
RLM_ACT_BAD_HOSTID (-1006)	No hostid supplied to the activation server.	This is an error in the configuration of your software
RLM_ACT_BAD_HOSTID_TYPE (-1007)	No acceptable hostid type transmitted to activation server	By default, activation will only accept hostids of types: 32-bit, ethernet, disksn, rlmidN, or ISV-defined. In addition hostids with value 0 are rejected as is 32-bit hostid 7f0100 (the default linux hostid). In addition, both the product definition and the activation key can specify the list of acceptable hostid types, you may not have passed one of these types. Finally, this error will also occur if there is an illegal character in a valid hostid type.

RLM_ACT_CANTREAD_DB (-1010)	Cannot initialize MySQL or connect to the database	Make sure that the isvname.mysql file in the cgi-bin directory contains the correct server name/port/username/password.
		
		This error can also result from:
		*(*1) a missing column in the "licf" table when attempting to revoke a license, or
		(2) a missing or empty or bad generator settings file in the case of RLM Hosted Activation.
		
		To debug this problem (v9.3 or later), once you are sure you have a good generator settings file, set the environment variable "RLM_ACT_SQL_DEBUG", then run the activation generator as follows:
		
		`ISV_mklic << EOF`
		`akey=1&hostid=2`
		`EOF`

Status return	Problem	Solution
		The generator will then print (to stdout) the SQL error status.
RLM_ACT_CANT_WRITE_FULFILL (-1011)	Cannot write the license key fulfillment data.	This is an internal server error. Check to make sure the MySQL server is running correctly.
RLM_ACT_CLIENT_TIME_BAD (-1012)	Time on client machine is more than 7 days off from server machine.	Correct the time difference between client and server machines (must be < 7 days)
RLM_ACT_TOOMANY_HOSTID_CHANGES (-1014)	Refresh-type activation has had it's hostid changed too many times.	
RLM_ACT_BLACLISTED (-1015)	Activation attempted from a host which is blacklisted for this server	Update the blacklist if it is not accurate
RLM_ACT_NOT_WHITELISTED (-1016)	Activation attempted from a host not on the whitelist for this activation key	Update the whittelist if it is not accurate
RLM_ACT_KEY_EXPIRED (-1017)	The activation key itself has expired	This would be a normal error for your customer who attempts to use an expired activation key.
RLM_ACT_BAD_GENERATOR (-1020)	Generator settings file missing or empty. License keys in generator (ISV.gen) bad or expired, or the generator is missing.	Make sure that the <isv>.gen file is present and correct (it will be 0-length on install until replaced). If you built your own generator, check to make sure it was built with a correct license_to_run.h
RLM_ACT_NO_KEY_MATCH (-1021)	Activation Key can't be found in database	This would be a normal error for a customer who uses an incorrect activation key
RLM_ACT_DB_READERR (-1026)	Error reading data from MySQL	The activation key table has the wrong number of columns (this can happen if you use a v9.1 database with the v9.2 software). The product definition table has the wrong number of columns.
RLM_ACT_GEN_PARAM_ERR (-1027)	Error in generator call to rlm_sign_license()	The license string passed into rlm_sign_license() is incorrect, most likely because something in your extra license parameters is invalid.
RLM_ACT_UNSUPPORTED_CMD (-1028)	Unsupported command	This is an error that a non-Activation PRO server returns when an advanced command is issued to it. This error also results in ActPro when attempting to activate a multi-license product with a REFRESH activation key.
RLM_ACT_REVOKE_TOOLATE (-1029)	Revoke attempted after license expiration. This is a warning – the rehostable hostid has	Licenses cannot be revoked after they expire. If you receive this error, you can reactivate the license if there is activation count remaining in the activation key. The

Status return	Problem	Solution
	been deleted.	activation server will NOT have returned count to the activation key, however, the rehostable hostid will be deleted.
RLM_ACT_KEY_DISABLED (-1030)	Activation Key Disabled	This results from an rlm_act_info() or rlm_act_keyvalid() call if the activation key has been disabled.
RLM_ACT_KEY_NO_HOSTID (-1031)	Key not fulfilled on this hostid	This results from an rlm_act_keyvalid() call if the activation key has not been fulfilled on the specified hostid.
RLM_ACT_KEY_HOSTID_REVOKED (-1032)	Key revoked on this hostid	This results from an rlm_act_keyvalid() call if the activation key has been revoked on the specified hostid.
RLM_ACT_NO_FULFILLMENTS (-1033)	No fulfillments to revoke	This results from an rlm_revoke() call if the activation key has no fulfillments to revoke. This can be treated as a warning.
RLM_ACT_LICENSE_TOOBIG (-1034)	Generated license exceeds maximum size.	A product definition that specified multiple licenses created a license that exceeds RLM_ACT_MAX_LICENSE characters.
RLM_ACT_NO_REHOST (-1035)	Rehostable hostids not supported for this product.	A rehostable hostid cannot be used for a counted license, and it also cannot be used for any product definition that specifies multiple licenses.
RLM_ACT_NO_CLEAR (-1038)	Unencrypted requests not allowed on this server	The activation pro server does not allow unencrypted requests. Set in the admin/generator settings tab.
RLM_ACT_KEY_TOOMANY (-1042)	Request for more licenses than remain on the activation key	Request fewer licenses.

Errors detected and returned locally by rlm_act_request() or by the HTTP server:

Status return	Problem	Solution
RLM_EH_NET_INIT (-103)	Can't initialize the networking code.	Most likely the URL specified in the call is incorrect.
RLM_EH_NET_WERR (-104)	Cannot write data to network.	Most likely the URL specified in the call is incorrect.
RLM_EH_NET_RERR (-105)	Can't read returned message from network.	If the write succeeds and the read fails, there are several possibilities: • the MySQL server is down or you have specified the wrong port, • The server, when it does the reverse DNS lookup on the client system, times out. This will be noted in the debug log as of v12.1 • Your connection might be slow.

Status return	Problem	Solution
		Activation uses a 10-second timeout by default, try setting the environment variable RLM_ACT_TIMEOUT to 45 or 60 to get a 45 or 60-second timeout, or • there is a proxy server in the middle. Try setting HTTP_PROXY.
RLM_EH_BADPARAM with RLM_ACT_BP_TOOMUCH sub-error (-123)	Request is too long to communicate.	Check the length of the extra license parameters, the hostid list, and the log data. Each of these should be < 80 bytes.
RLM_EH_CANTCONNECT_URL (-132)	Can't connect to the specified URL.	Check that the URL is correct and that the activation server is installed in cgi-bin. Pass the URL without the trailing /cgi-bin, e.g. http://www.reprisesoftware.com, NOT http://www.reprisesoftware.com/cgi-bin
RLM_EH_ACT_BADSTAT (-134)	rlm_act_request() received no status in the returned message from *ISV_mklic*	This is most likely a configuration error on the server side, which causes the *ISV_mklic* process to return an incomplete reply. The HTTP server error logs might well indicate the problem. If your server is known to be running correctly, this could be caused by a proxy server that isn't configured correctly. This can also be caused by the Avast anti-virus software - which interrupts the return from the activation server and returns a partial buffer, causing this error. The workaround is to turn off the Avast "Web Shield". This can be turned off for 10 minutes in the Avast GUI.
RLM_EH_BAD_HTTP (with sub-error equal to HTTP status) (-136)	Other HTTP error	See the full error message for the HTTP status. Use RLM's extended error messages. For some HTTP 404 errors, we have encountered Apache installations that do not make cgi-bin files executable by default.
RLM_EH_UNLICENSED (-143)	RLM license keys missing or invalid	Check your license_to_run.h include file and ensure your RLM keys are valid.
RLM_EH_ACT_OLDSERVER (-145)	Activation server version too old.	This happens when a pre-v9 activation server is used with a v9+ client which encrypts the request. This can also happen if there is a proxy server installed at the user site.
RLM_EH_CANT_CREATE_REHOST (-151)	The rehostable hostid for this product could not be created.	Verify that the user has write access to either /var/tmp/{isvname} on unix/mac systems or "documents and settings/all

Status return	Problem	Solution
		users/application data/reprise" on Windows
RLM_EH_REHOST_TOP_DIR_EXISTS (-152)	The directory which contains a rehostable hostid for this product already exists, so it cannot be created.	Verify that this product has not already been activated with a rehostable hostid. If it has, it cannot be activated again on the same machine.
RLM_EH_REHOST_EXISTS (-153)	The rehostable hostid already exists for this product, so it cannot be created.	Verify that this product has not already been activated with a rehostable hostid. If it has, it cannot be activated again on the same machine.
RLM_EH_NO_FULFILLMENTS (-154)	No fulfillments exist to revoke	An attempt was made to revoke a license which had a zero fulfillment count on this server. This can happen if the revoke operation is attempted after already being revoked.
RLM_ACT_BAD_REDIRECT (-1013)	HTTP redirect does not specify hostname, or can't connect to specified hostname.	This is a webserver redirect error – no redirected hostname was specified, or we can't connect to the redirected hostname.
RLM_ACT_NO_PERMISSION (-1018)	HTTP server did not allow request to be processed.	This is a configuration error – your license generator cannot run on the webserver.
RLM_ACT_SERVER_ERROR (-1019)	HTTP request returned HTTP status 500 - INTERNAL_SERVER_ERROR	The activation server is not configured correctly. Look at the web server error log files to find the source of the error. Without this information, it is impossible to diagnose the error.
RLM_ACT_NO_AUTH_SUPPLIED (-1022)	Proxy requires authentication	Supply proxy credentials via the HTTP_PROXY_CREDENTIALS environment variable.
HTTP_PROXY_AUTH_FAILED (-1023)	Proxy credentials didn't work	Proxy requires authentication and the credentials supplied failed.
RLM_ACT_NO_BASIC_AUTH (-1024) RLM_ACT_BAD_URL (-1036)	Proxy does not support BASIC authentication License generator not found at this URL	RLM activation uses only basic authentication with proxy servers. The URL in the rlm_activate() call is incorrect, or your license generator is not present or not executable.

Appendix I – Refresh-type activations (deprecated)

Rehosting via Activation Refresh (deprecated)

Note: the REFRESH activation type has been deprecated in RLM v9.3. Please use the new rehostable hostid type if you wish to allow users to transfer licenses from one system to another.

Activation can be set up to support rehosting of licenses by the end user without ISV involvement. This involves the use of "refresh" licenses, which are normal RLM licenses, except they are typically issued for a very short duration - on the order of days. Refresh licenses are reactivated frequently, either by a generic utility supplied with RLM, or by a utility supplied by the ISV, using the RLM refresh API. Each time a refresh license is reactivated, it gets a new (later) expiration date, some number of days from the reactivation date specified by the ISV.

Rehost and Refresh Concepts

Note: the REFRESH activation type has been deprecated in RLM v9.3. Please use the new rehostable hostid type if you wish to allow users to transfer licenses from one system to another.

A rehostable license is one that is intended to be reactivated frequently and receive a new expiration date with each reactivation. Rehostable licenses are typically of very short duration. The main purpose of the refresh activation type is to support rehosting of licenses. Rehosting is when your customer wants to move a license from one system to another. Traditionally, the ISV had no way of knowing if the customer ceased running the application on the original system after issuing a license for the new system. With very short term rehostable licenses, the ISV is assured that the maximum period during which the end user can run the licensed application on both the old and the new systems is the (short) duration of the rehostable license. Refresh-type licenses must be node-locked, they can never be floating.

For example, if the ISV specifies that the rehostable license for a product should expire 4 days after activation, a rehost of that product will effectively shut off use of the product a maximum of 4 days after the rehost. If the refresh operation at the end user site is run daily and it fails for some reason on a given day, the license is still good for another 3 days - enough time to resolve the refresh failure.

Rehostable licenses also give the ISV a way to revoke a license granted to a customer should that user fail to meet contractual obligations, for example. The ISV can simply disable refresh of the user's license on the activation server, and refresh attempts of that license will fail from that point forward.

The client side of refreshing is automated, so it can be performed daily without an undue burden on the customer. Reprise supplies a "Refresh API" for license refreshing, which the ISV can use in a custom-developed refresh utility. Reprise also supplies a generic refresh utility which can be supplied to the customer by the ISV. The generic utility, "refresh_util", is meant to be set up to run daily as a Scheduled Task on Windows or as a cron job on Unix.

Rehostable Licenses

A license is rehostable if the activation key that was used to activate it is a refresh type activation key. This is set in the rlc page that creates activation keys. The license generator does a couple of special things when it activates refresh licenses:

- It includes the optional attribute "_refresh_url". This is the URL that the refresh utility contacts to refresh the license, that is, the ISV's activation server. This is specified by the ISV in rlc, in the "Setup RLC" section. The activation server includes the refresh URL in every refresh license it creates.
- It includes the optional attribute "_refresh_id", which is the activation key for refreshing the license. This is also used by the activation utility.
- It includes a the options attributes "start_date" and "replace", which are used internally by the rlm library to obsolete previous instances of the same license.

NOTE: At the time of this release (v7.0), the refresh utility supports only nodelocked, uncounted licenses.

Refresh Utility

Reprise supplies both a stand-alone utility (isv_refresh) that can be used as-is at the customer site to refresh licenses, and an API (Appendix A – Activation API on page 57) around which the ISV may build a custom refresh utility.

rlmutil rlmrefresh takes an argument specifying the directory in which it should process rehostable licenses, and an optional argument specifying the name of an ISV whose licenses are to be refreshed. Without the ISV argument isv_refresh processes rehostable licenses from all ISVs in the specified directory. It prints details about any errors it encounters while refreshing the licenses, unless the optional -q argument is specified. On Windows a batch script can be set up to supply arguments and also to redirect the output to a file. It can be run manually, but it is intended to be run automatically as a cron job on Unix or a Scheduled Task on Windows. The norm is to run it once a day.

In order to avoid all customers refreshing their licenses at the same time every day and overloading the ISV's activation server, the ISV can request that each customer run the refresh utility at an assigned time every day. A custom refresh utility could randomize the refresh time.

Rehosting

Rehosting a license with refresh is a matter of turning off the refreshing of it on the original host, and activating it on the new host. Disable refresh by disabling the refresh of the directory containing the license if refresh_util is being used. Then activate the license on the new host, and enable refreshing of the license on the new host.

Product Definition and Activation Key Creation for Rehostable Licenses

- Define the refresh URL in rlc. Select "Setup RLC" from the menu on the left, and set the URL of the Activation server to contact for refreshes (your activation server name, eg, www.reprisesoftware.com). You only have to do this step one time.

- Define the product in rlc. The product must be defined as a nodelocked license (as of RLM v7.0). Choose a short duration for the license in the expiration date field. You do this each time you set up a rehostable product.
- Create Activation Keys for the product in rlc. Under "Activation Type", choose "Refresh". Set the number of rehosts allowed to the quantity desired. You create one activation key for each customer who has purchased this product.

Appendix J – Redirecting Your Activation Website

When you use RLM activation, your application contains the URL of the activation server. If you are running the activation server at your own website, it may be fine to hard-code this URL into your software.

However, if you think that there may be a time when you might want to move the website, there are a couple of options you can employ to make this change more easily:

1. you could create some kind of configuration which your software reads at runtime to decide which URL to use to activate, or
2. You can use the ISVNAME_ACT_URL environment variable to override the activation URL in the rlm_activate() call (see the description of rlm_activate() in the Reference Manual), or
3. you can set up your URL to redirect to a different (presumably newer) URL.

If you use Reprise's hosted activation service, Reprise strongly recommends that you employ one of these techniques. Obviously, the first two techniques are more flexible, because they would allow you to continue operating even if the original URL goes away completely. This is what Reprise Software recommends.

If you wish to redirect your URL, you can do this with an .htaccess file if you are using a linux server with Apache:

Set up a new domain which you use as your activation URL and redirect the whole domain to the actual activation domain. To do this, follow these steps:

- Create the domain (usually a subdomain, e.g. activation.yourcompany.com).
- In the top-level of the domain, create a .htaccess file with the following contents:

```
Options +FollowSymLinks
RewriteEngine on
RewriteRule (.*) http://www.actual-activation-domain.com/$1 [R=301,L]
```

(Note: Change “actual-activation-domain” to your domain name. Also, this .htaccess method of redirection works ONLY on Linux servers having the Apache Mod-Rewrite module enabled.)

(Note: if you want to redirect just the cgi-bin directory, you will need to remove/comment out the “ScriptAlias” line from your apache conf file (…../apache/conf/httpd.conf).

Appendix K – Automatically Blacklisting IP addresses

If someone writes a program to guess activation keys, they will present a large number of invalid activation keys to the server in a short period of time. RLM Activation Pro v9.4 has the capability to automatically blacklist this IP address based on parameters you specify.

The first time an invalid activation key is presented to the license generator by a particular IP address, the license generator logs the time. After this, if N bad keys are presented within a specified time interval, the IP address is automatically blacklisted. Once this time interval passes without N bad keys being attempted, the whole process resets (in other words, N bad attempts will be required before the IP address is blacklisted).

By default, the interval is 600 seconds (10 minutes) and the number of bad keys which cause the IP address to be blacklisted is 40. However, you can change both of these parameters in the "Database" tab within the "Admin" tab (available to admin users), near the bottom of the form:

In this example, 66 bad activation keys within 600 seconds will cause the IP address to be blacklisted. To set the parameters, enter the 2 parameters into the text boxes and press "Set Blacklist Parameters".

Once blacklisted, the IP address will appear in the normal blacklist. The blacklisted IP address will have to match the IP address exactly, rather than the substring domainname match of normal blacklisted hosts.

Appendix L – How Activation Pro chooses a hostid

RLM Activation Pro will chose a hostid for the generated license based on the input from the RLM-licensed application.

If your application does not specify a hostid to the *rlm_act_request()* or *rlm_activate()* call, then the RLM client library supplies the standard set of hostids for the machine on which the software is running. Alternately, you can specify a hostid-list in the rlm_act_request() call, or via the *rlm_act_set_handle()* call if you use *rlm_activate()*.

Prior to v11.0, rlm would only activate licenses with rehostable, non-zero RLM_HOSTID_32BIT, RLM_HOSTID_ETHER, RLMIDn, RLM_DISKSN, or ISV-defined hostids. Any other hostid will return an RLM_ACT_BAD_HOSTID_TYPE status from *rlm_act_request()*. (Note: ISV-defined hostids were added to the list of legal hostids in RLM v4.0). Beginning in v11.0, you can specify exactly the hostids you will accept with the *rlm_isv_cfg_actpro_allowed_hostids()* call in rlm_isv_config.c, then either re-building your license generator or creating a new generator settings file. See the RLM Reference Manual "Customizing RLM with rlm_isv_config" section in the "Integrating RLM Into Your Product" chapter for more details.

The priority is (assuming the particular hostid type is enabled):

- rehostable hostid
- ISV-defined hostid
- rlmid hostid
- Alternate Server Hostid
- Disk Serial Number
- ethernet address
- 32-bit hostid
- ip address hostid
- user-based hostid
- host-based hostid
- serial number hostid
- string hostid
- DEMO hostid
- ANY hostid
- If none of the hostid types above are present (or enabled), the activation software will return RLM_ACT_BAD_HOSTID_TYPE.

You can override RLM's notion of the hostid and specify an exact hostid to be used by the activation server by calling *rlm_act_set_handle()* with the RLM_ACT_HANDLE_HOSTID_LIST parameter (or passing the hostid-list parameter to *rlm_act_request()*). The *hostid_list* parameter can contain a list of hostids for use in nodelocked licenses. This is specified with the following syntax:

```
list:list-of-hostids
```

For example:

```
list:user=joe host=sam ip=192.16.7.23 3f902d8b0027
```

If a list is supplied, note the following:

- The activation software uses the hostids in the list as you specified, even if they are not enabled or “Secure”.
- If the license to be activated is a served license (floating), only the first hostid in the list is used.
- The number of available activations on the activation key is decremented by 1 regardless of the number of hostids in the license created.
- The hostid list must be less than RLM_ACT_MAX_HOSTID_LIST characters long (205) including the “list:” prefix.
- The hostid list can contain no more than RLM_MAX_HOSTID_LIST (25) hostids.

  This capability can be used to create a license which works on 2 (or more) systems, e.g. to create a license for a primary and a backup system. It can also be used to pass a hostid of a less secure type, or which is not enabled by default, to be used, e.g. the *hostid-list* "list:ip=172.16.7.12" will cause the activation software to use the IP address as a hostid without returning RLM_ACT_BAD_HOSTID_TYPE.

Appendix M – Using custom license generators

In some situations, you will want to develop your own license generator, to replace the standard RLM license generator. You can do this with Activation Pro.

To do this, take the following steps:

1. Define your license generation algorithm in the product definition. 0 is the standard RLM algorighm, any positive integer is defined by you. This integer will be passed to your license generator in step 2.
2. Create a function called "rlm_ext_sign_license() similar to the example at the end of this section, except that you will parse the input LICENSE line and re-write the license in whatever (text) format you want.
3. Link your rlm_ext_sign_license() function into the license generator, placing your object file before the library so that your generator is used in place of the dummy function in the library. To build your license generator, follow the instructions for building your license generator as if you had an ISV-defined hostid. The instructions to do this are on page 19.
4. Install your license generator (ISV_mklic or ISV_mklic.exe) and you are ready to go.

The example rlm_ext_sign_license() function is here:

```
/*****************************************************************************

        COPYRIGHT (c) 2007, 2013 by Reprise Software, Inc.
       This software has been provided pursuant to a License Agreement
       containing restrictions on its use.  This software contains
       valuable trade secrets and proprietary information of
       Reprise Software Inc and is protected by law.  It may not be
       copied or distributed in any form or medium, disclosed to third
       parties, reverse engineered or used in any manner not provided
       for in said License Agreement except with the prior written
       authorization from Reprise Software Inc.

 *****************************************************************************/
/*
 *      Function:       stat = rlm_ext_sign_license(rh, algorithm,
 *                                                  server_hosid, license)
 *
 *      Description:    Create a license - ISV-defined
 *
 *      Parameters:     (RLM_HANDLE) rh
 *                      (int) algorithm - algorithm to use (ISV-defined)
 *                      (char *) server_hostid - Server hostid, if applicable
 *                      (char *) license - input/output license string
 *
 *      Return:         (char *) license - input/output license string
 *                      (int) status - status return
 *
 *      M. Christiano
 *      7/27/12
 *
 *
 *      This function should be replaced by the ISV with an equivalent
 *      function which creates the license in the string "license".
 *
 */
```

```
#include "license.h"

int rlm_ext_sign_license(RLM_HANDLE rh, int algorithm, char *server_hostid,
                              char *license)
{
/*
 *      If this were the real function, the string "license" would
 *      be replaced by the actual generated license.
 *
 *      Note that "license" has RLM_MAX_LINE allocated bytes only,
 *      and is allocated on the stack.
 *      In practice, however far fewer bytes can be transmitted
 *      back to the client, since the whole HTTP message can be
 *      no more than 1024 bytes.
 *
 *      Return 0 for success, a non-zero error code using the status
 *      in license.h for an error.
 */
       (void) strcpy(license, "dummy externally-signed license\n");
       return(0);
}
```

Appendix N – Bulk Loading Customer Data

It is possible to load customer data from an external source into Activation Pro. To do this, create a tab-delimited file with all the customer data. Once you have this file, in the Admin tab, under "Database", you will see "Bulk-Load Customer database data" at the bottom of the form. Browse to your file, then press "Upload File". Your customer data will be loaded.

Reprise STRONGLY suggests you upload a file with one customer first, so that you can verify your upload data format. If something goes wrong, you can delete this customer and try again.

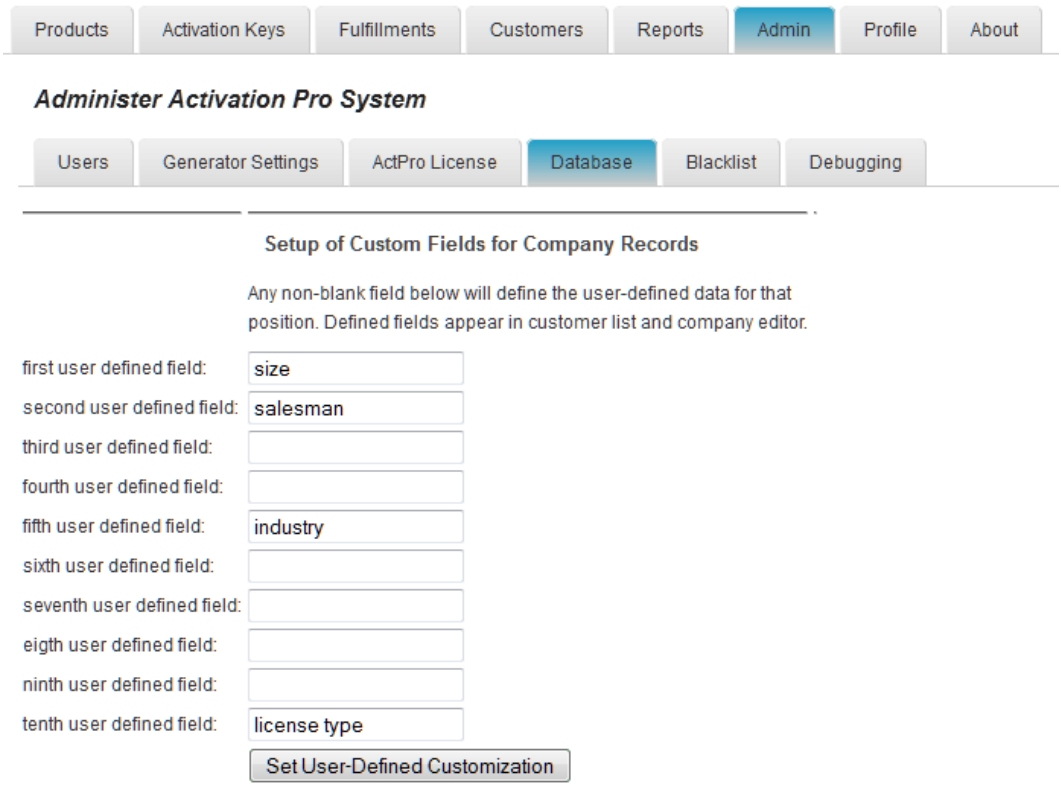
NOTE: This file MUST be tab-delimited, and it MUST contain all 26 fields below, in the order below. Only the contact name and email are required to be non-empty:

Data Item	meaning	data type
contact	Contact person name	string (60)
contact_type	Contact type (types defined in the contact_types table)	string (60)
title	Job Title	string (60)
phone	Phone #	string (20)
fax	Fax #	string (20)
email	Email address	string (60)
notes	General notes	string (60)
company	Company name	string (60)
addr1	Address line 1	string (60)
addr2	Address line 2	string (60)
addr3	Address line 3	string (60)
city	City	string (60)
state	State	string (60)
zip	Zip/Postal code	string (60)
country	Country	string (60)
Notes	Company notes	string (255)
u1	Company User-defined	string (60)
u2	Company User-defined	string (60)
u3	Company User-defined	string (60)
u4	Company User-defined	string (60)
u5	Company User-defined	string (60)
u6	Company User-defined	string (60)
u7	Company User-defined	string (60)
u8	Company User-defined	string (60)
u9	Company User-defined	string (60)
u10	Company User-defined	string (60)

Appendix O – Customizing the Customer Data

The list of companies in Activation Pro contains a fixed set of data: Company Name, Address lines 1-3, City, State, Zip, Country. You can add up to 10 additional fields of your choice to this data. Any additional fields you add will appear in the Customer Editor as well as in the list of customers.

To customize the data, select the "Database" tab from the "Admin" tab. At the bottom of the form, press "Customize Company fields". You will see the following form:



When you have added the fields you want, press the "Set User-Defined Customization" button at the bottom.

Note that you can put values in any of the 10 user-defined fields. The strings you put on this form will appear in the Company Editor and will also be the headings on the customer list (so you probably want to keep them relatively short). For example, with the customization above, the company editor will display the extra "size", "salesman", "industry" and "license type" prompts.

Appendix P - Importing Activation Keys to ActPro

On occasion, you might want to create activation keys outside of Activation Pro and then import those keys automatically into ActPro. You might do this, for example, if you wanted your order processing system to create an activation key for your customer's purchase, then populate ActPro with this activation key.

The activation pro kit contains an example program to accomplish this. This program is called *import_key.c* and it can be used as an example to write your own code, or it can be used as-is as a command-line utility.

Usage of *import_key* is as follows:

% import_key activation-key product-id [-c count] [-t n|r] [-e key-exp-date] [-l lic-exp-date] [-w whitelist] [-o other-params] [-n notes] [-z contact-id]

The product-id and activation-key are both required.

To determine the product-id, view the product definitions in the Activation Pro GUI, and hover over the Product name in the list. The product-id will be displayed. The activation-key can be any printable string of less than 60 characters.

To determine the person's contact-id, view the customer list in the Activation Pro GUI, and hover over the contact name in the list. The contact-id will be displayed.

All parameters correspond to the parameter in the "Create Activation Key" GUI.

The -t parameter specifies whether this key is a "normal" (n – the default), or ReActivate key (r).

To build *import_key*, you must have the MySQL connector kit installed (instructions for this are in step 4 of the Activation Pro Setup chapter. To compile and link import_key (where ***sql*** is the directory where the mysql connector kit is installed):

cc -c import_key.c -I***sql***/include

cc -o import_key import_key.o ***sql***/lib/libmysqlclient.a -lm

NOTE: As an alternative to this method, you can use Web Services to import activation keys into the ActPro database. See the next section.

Appendix Q - Using Web Services to Access the ActPro Database

NOTE: An earlier version of ActPro web services is deprecated as of this release. It is still present in ActPro but is not documented here, and it has been renamed "actpro_web_serv_11.2.php". See documentation for earlier versions of ActPro for that version of ActPro web services. The current version of ActPro web services is actpro/actpro_web_serv.php, and is described below.

ActPro supports Web Services to access the ActPro database. This is implemented as a simple REST-style service, using JSON to encode the information going into and out of the database. The following HTTP commands are used:

- POST: Create one or more rows
- GET: Fetch one or more rows
- PUT: Update one or more existing rows
- DELETE: Remove one or more existing rows

Tables that can be accessed, and the operations that can be done on them are as follows:

- company: post, get, put, delete
- contact: post, get, put, delete
- prod: post, get, put, delete
- keyd: post, get, put, delete
- keyf: get, delete
- licf: get, delete

Using ActPro Web Services:

- Data is transferred in JSON encoding (see below).
- https should be used in all web services transactions, to keep data secure.
- User authentication uses the same usernames and passwords as the ActPro user interface does. Credentials are passed via HTTP Basic authentication.
- actpro/actpro_web_serv.php implements the web services on the ActPro server.
- Use of ActPro web services can be prototyped easily with the curl command line tool.

The following example curl command **POST**s myjson.json to the ActPro server **actpro.mycompany.com** using username **joe** and password **joesp@ssw0rd**.

```
curl -X POST -H "Content-Type: application/json" -H "Accept:
application/json" -d @myjson.json
https://actpro.mycompany.com/actpro/actpro_web_serv.php -u
'joe:joesp@ssw0rd'
```

JSON format for data sent to ActPro Web Services

The structure of the JSON content sent to ActPro web services consists of a header object and where data is being provided, an array of data objects. For example, if the message posts data to the database.

```
{
	"header": <header object>,
	"data":[ <array of data objects>]
}
```

The header object always specifies a username and MD5 hashed password, and a table to operate on. The operation is specified externally, in the http/s header. Some operations may also specify query information. For example, this specifies the row in the **licf** table whose akey is **key12:**

```
{
	"header":{
		"user":"webservicesuser",
		"password": "5fd0de2763a715d56ad571513a389192",
		"table":"licf",
		"akey":"key12"
	}
}
```

Note that the username and password specified here are different than the username and password specified for above HTTP basic authentication. Those credentials must match what's in the .htaccess file, if any, in the ActPro directory on the activation server. The username and password specified in the header must match a username and password in the ActPro database, ie, what you would use to log in to the ActPro UI. The password is supplied as an MD5 hash. If you need to create an MD5 hash of your ActPro account password manually, the web site http://www.danstools.com/md5-hash-generator is useful.

NOTE: The user specified in the header must have *admin* or *edit* access in order to interact with the ActPro database. A *view* or *portal* access user will receive an insufficient privilege error.

JSON for POST operations includes the data to post. Here is JSON specifying 2 rows to be added to the **keyd** table:

```
{
	"header":{
		"user":"webservicesuser",
		"password": "5fd0de2763a715d56ad571513a389192",
		"table":"keyd"
	},
	"data":[
		{
			"akey":"sam",
			"active":"1",
			"product_id":"13",
			"count":"0",
			"type":"0",
			"rehosts":"0",
			"lastdate":"",
			"exp":"",
			"kver":null,
```

```
                "kver_type":"0",
                "white":"",
                "misc":"",
                "notes":"",
                "contact_id":"0"
        },
        {
                "akey":"bill",
                "active":"1",
                "product_id":"13",
                "count":"0",
                "type":"0",
                "rehosts":"0",
                "lastdate":"",
                "exp":"",
                "kver":null,
                "kver_type":"0",
                "white":"",
                "misc":"",
                "notes":"",
                "contact_id":"0"
        }
    ]
}
```

If you want ActPro Web Services to create the name of the activation key (akey column), don't specify a value for akey in the JSON input. Web Services will create the name, using the same algorithm RLC uses when you create an activation key and don't specify the name. That is, an activation key name of the form xxxx-xxxx-xxxx-xxxx will be created. Web Services returns the names of the activation keys it created. (See Return Data Format below)

JSON for PUT operations (modifying an existing row or rows) consists of the header and the data to be modified. Here is JSON specifying a new value for the "add_akey" column for any row in the prod table whose value for the "product" column is "simulator":

```
{
    "header":{
        "user":"webservicesuser",
        "password": "5fd0de2763a715d56ad571513a389192",
        "table":"prod",
        "product":"simulator"
    },
    "data":[
        {
            "add_akey":"1"
        }
    ]
}
```

JSON for GET and DELETE operations specifies only the header, and not any data. Here is an example that specifies rows in the contact table whose company_id is 161. This might be used with either a GET or a DELETE:

```
{
    "header":{
        "user":"webservicesuser",
```

```
            "password": "5fd0de2763a715d56ad571513a389192",
            "table":"contacts",
            "company_id":"161"
      }
}
```

Complex queries

More complex queries can be specified in the header. Previous examples have been in the form "<column>":"<value>". When searching the database, this turns into an SQL WHERE clause "WHERE <column> REGEX <value>". REGEX is the regular expression operator and without special characters it searches for the string supplied. <value> can also be specified as <[<operator>] <value>>, where <operator> is one of ">", ">=", "=" "<=", "<", and "<>" (not equal).

For example, to search for rows in the keyd table whose value for the count column is 10 or more, specify:

```
"count":">= 10"
```

Multiple header rows specifying columns are ANDed together. So for example, to search for rows in the licf table whose akey is "1234-5678" and whose hostname is "blue", specify this in the header:

```
"akey":"1234-5678",
"hostname":"blue"
```

Finally, if a query beyond the capabilities outlined above is necessary, you can specify any arbitrary SQL WHERE clause as "sql_where":"<where clause>", for example,

```
"sql_where":"`count`=1 or `count`=2"
```

Return data format

Data returned from ActPro web services is also encoded in JSON, as follows:

```
[
      { header object }
      [ <array of data objects> ]
]
```

The header object contains a status "OK" or "ERROR", and optionally a message. For example here is the response to the get of a non-existent row:

```
[{"status":"OK","message":"0 rows found"},[]]
```

Here is the response to a get of all rows in the licf table which returned 2 rows (some columns left out for brevity):

```
[
      {"status":"OK","message":"2 rows found"},
      [
            {"id":"20727","akey":"akey345","product_id":"1",<omitted columns>},
            {"id":"20728","akey":"test_key","product_id":"8",<omitted columns>}
      ]
]
```

POST operations on tables with auto-increment primary keys (company, contact, prod) return the key values for the rows created, for example,

```
[{"status":"OK","message":"3 rows added, IDs: 62 63 64 "}]
```

POST operations on the keyd table where the names of the activation keys are not specified in the JSON input return the activation key names for the rows created. For example:

```
[{"status":"OK","message":"3 rows added. Activation keys: 4662-
6168-1884-8782 4664-6168-7701-1435 4665-6168-2611-4584 "}]
```