License Manager

License Manager

version 1.3.0

About this Plugin

This plugin facilitates secure communication between the License Module and the client software, acting as a license server for both the provisioning and validation of client installations.

Supported Protocols

- **RSA Digital Signature**

  Currently the only supported protocol. RSA Digital Signature encrypts messages with 256-bit AES, and signs them with 1024-bit RSA signature.

Using the Plugin

Configuration

To configure the pulgin visit [Settings] > [Company] > [Plugins] > [Installed] and click the **Manage** button on the License Manager.

- **Log File**

  This is the full path on the server to the file that will log incoming and outgoing license request data.
- **Log File (Anonymous Trials)**

  This is the full path on the server to the file that will log trial requests. This log is used by the system only when provisioning Anonymous Trials, and is used only to ensure that duplicate trials are not provisioned for the same domain or IP.
- **Log File Format**

  The format to use when adding data to the log file (does not affect the anonymous trials log file).

  - **%1$s** - The log ID
  - **%2$s** - The direction (input or output)
  - **%3$s** - The date in ISO 8601 format
  - **%4$s** - The client IP address
  - **%5$s** - The requested URI
  - **%6$s** - Data

Configuring Trials

Two types of trials are supported:

- **Client Trials**

  These require that the user have or create a client account when ordering a service that uses the License Module. The service will be added to the client's account. To configure this type of trial, simply create a package with a price of 0 (if free), with the cancel at end of term option selected. This will ensure that the service does not renew and is cancelled immediately.
- **Anonymous Trials**

  Anonymous trials are always provisioned to the same client account, and can only be configured for a single package price point.

  To request an anonymous trial license key, simply POST the 'domain' to https://domain.com/path_to_blesta/plugin/license_manager/trial.

  Settings that control anonymous trial options are found in ./config/license_manager.php

  - **LicenseManager.trial_length** - A string representing the length the trial is active (default "30 days"). When a trail is provisioned it will automatically be scheduled for cancellation date based on this setting.
  - **LicenseManager.trial_client_id** - The client ID all trials will be provisioned under.
  - **LicenseManager.trial_package_pricing_id** - The ID of the package pricing ID that identifies the package for anonymous trials.
  - **LicenseManager.trial_package_group_id** - The ID of the package group that the pricing ID belongs to.

How to get the Package Pricing ID?

To get the Package pricing ID the easiest way, simply inspect element on the pricing table when editing a package. For example:

```
<td class="medium">
        <input type="hidden" name="pricing[id][]" value="77" class="pricing_id" wfd-id="195" kl_vkbd_parsed="true"
>
        <input type="text" name="pricing[term][]" value="14" class="term stretch" wfd-id="194" kl_vkbd_parsed="
true">
</td>
```

So our pricing_id value is *77*

How to get the Package Group ID?

To get the Package Group ID, you can do this by going to Packages > Package Group and edit the group in question like: `https://blesta.com/admin/packages/editgroup/10/` The number *10* is the Package Group.

Code Integration

Initialize the License Manager

Before you can begin using the License Manager within your code you have to initialize it with some data.

```
require_once "path/to/license.php";

$server_url = "https://domain.com/path_to_blesta/plugin/license_manager/validate/";
$license_key = ""; // The client's license key
$public_key = ""; // The client's public key (if they have one)
$shared_secret = ""; // A random shared secret value that exists for this Licese Module product
$path_to_phpseclib = dirname(__FILE__) . DIRECTORY_SEPARATOR . "phpseclib"; // The path to the phpseclib library

$license = new License($path_to_phpseclib);
$license_manager = $license->getManager();
$license_manager->setLicenseServerUrl($server_url);
$license_manager->setKeys($license_key, $public_key, $shared_secret);
```

IMPORTANT You'll note that we included license.php in the above sample, but in reality you will likely want to copy/paste the code in license.php into a core file within your project, to prevent tampering.

Request Public Key

After you've obtained a user's license key, you'll need to fetch the public key from the server so that you can send and receive other data. You should store the public key in the same manner in which you store the license key, either a flat file or in a database.

```
$public_key = $license_manager->requestKey();
```

Request License Data

To fetch data from the license server invoke the **requestData()** method. You should store the data returned exactly as it appears in your database or a flat file along with the license key and public key.

```
$product_version = "1.0.0"; // The version of your product
$custom_data = array('version' => $product_version); // An array of anything you want to send to the license server. 'version' is the only custom field currently saved for the service
$license_data = $license_manager->requestData($custom_data);
```

Validate License Data

```
$ttl = 1209600; // Amount of time local license data should be valid for (60*60*24*14 = 1209600 = 14 days)
// Request license data if we don't already have some, or if what we have is no longer valid...
$license_data = $license_manager->requestData();
$data = $license_manager->validate($license_data, $ttl);

switch ($data['status']) {
    case "valid":
        // Everything looks good

        #
        # TODO: Do whatever you need to do when the license is valid (which is probably nothing)
        #

        break;
    case "invalid_location":
        // Invalidation installation location

        #
        # TODO: Do whatever you need to do when the license is invalid
        #

        break;
    case "invalid_version":
        // Invalid installation version

        #
        # TODO: Do whatever you need to do when the license is invalid
        #

        break;
    case "suspended":
        // License has been suspended

        #
        # TODO: Do whatever you need to do when the license is suspended
        #

        break;
    case "expired":
        // License has expired

        #
        # TODO: Do whatever you need to do when the license is expired
        #

        break;
    case "canceled":
        // License has been canceled

        #
        # TODO: Do whatever you need to do when the license is canceled
        #

        break;
    case "unknown":
        // License data does not exist or is corrupt

        #
        # TODO: Do whatever you need to do when the license data does not exist or is corrupt
        #

        break;
}
```