

What's new in QLM v16

.NET 6

- The QLM Management Console now uses .NET6 - Windows
- The QlmLicenseLib.dll is now available for .NET6.
- The QlmLicenseWizard.exe is now available for .NET 6 - Windows only.

QLM Management Console

- The tooltip over the license key in the Manage Keys tab now displays Analytics information such as the real Last Accessed Date.
- You can now create email notifications based on changes to Customers table such as when a new customer is created.
- Setting the SMTP Server on the client side now automatically updates the server's settings.
- Scheduled Tasks - You can now force the Start Time of a scheduled task to always run at a specified time.
- Scheduled Tasks - Performance and memory footprint improvement when sending automated emails.

QLM License Wizard

- User Registration Page - New Properties
- QlmTrialKeyUserGroup allows you to set the Affiliate ID when generating a trial key from the QLM License Wizard.
- QlmTrialKeyArgs allows you to set the any argument when generating a trial key from the QLM License Wizard.
- QlmTrialKeyUserData allows you to set the User Data when generating a trial key from the QLM License Wizard.

Library and API

- All QLM DLLs are now digitally signed.
- Added strong name validation of QlmLicenseLib.dll to the LicenseValidator class to prevent hacking DLL for non .NET applications that dynamically load the DLL. You must generate a new LicenseValidator class to take advantage of this feature.
- Added support for QlmUniqueSystemIdentifier2.
- New API GetUniqueCombinedSystemIdentifier that combines system identifiers.

License Server

- Added 2 server properties to configure the trial registration form title and caption.
- QLM Self Help now displays the detailed license information including remaining deactivations allowed and Analytics.
- New Server Property: BlockComputerIDs allows you to block specific computer IDs from getting activated.

New Requirements

- The QLM License Server now requires .NET 4.62.
- The QLM Customer Portal now requires .NET 4.72.

Important Notes

- EXE Protection will no longer be maintained.
 - QLM Engine Version 4.0 and lower are no longer supported.
-

Upgrade Procedure

To upgrade to QLM v16, you must first install the QLM Management Console on your client system by running the qlmsetup16.exe. To determine if you are eligible to a v16 upgrade, you may contact us or click the About tab. You can download QLM v16 from our [web site](#)

QLM License Server Upgrade

If we are hosting your QLM License Server, contact us to upgrade the License Server. If you are hosting your own License server, there are 2 ways to upgrade: in-place or in parallel. In-place upgrade will replace your existing server with the new one while the parallel upgrade allows you to run both servers in parallel until you are ready to switch. If you are upgrading from QLM v7+, the in-place upgrade is safe. If you are upgrading from earlier versions, we recommend the parallel upgrade.

In-Place Upgrade

1. Update the DLLs on your web server with the DLLs located in %Public%\Documents\Quick License Manager\DeployToServer\QlmLicenseServer\bin
2. If you have not executed the sql2005.aspnet.sql script when you created your QLM 5.x DB, this script is now required. The script is located in %Public%\Documents\Quick License Manager\DeployToServer\QlmLicenseServer\Db\sql2005.aspnet.sql. This step is not required if you were running QLM v7+.
3. Ensure the Application Pool associated to the QLM License Server is set to use .NET 4.0.
4. Once the License Server is updated, start the QLM Management Console, go to Sites and click on Upgrade Database Schema.
5. If you are using our eCommerce integration, do the following:
 - Go to the Manage Keys tab
 - Click on the Commerce Providers item in the toolbar
 - Select the eCommerce provider you are using
 - Modify the Dll field and replace the existing value with: QlmWebService.dll
 - Ensure that the eCommerce Provider that you are using is enabled.

Parallel Upgrade

To upgrade the QLM License Server, we recommend that you create a new virtual directory and install a QLM License Server in parallel to your existing QLM x.y License Server. Both License Servers can be configured to point to the same database. Once the QLM v16 License Server is configured and

properly running, you can remove the QLM x.y License Server.

1. Create a new folder called qlm16 on your web server in the same parent folder as your existing QLM License Server.
2. Copy the files located in %Public%\Documents\Quick License Manager\DeployToServer\QlmLicenseServer to qlm16
3. If you have not executed the sql2005.aspnet.sql script when you created your QLM 5.x DB, this script is now required. The script is located in %Public%\Documents\Quick License Manager\DeployToServer\QlmLicenseServer\Db\sql2005.aspnet.sql. This step is not required if you were running QLM v7+.
4. Copy the following web.config settings from your existing QLM License Server web.config file to the new one:
 1. connectionStrings
 2. communicationEncryptionKey
 3. adminEncryptionKey
 4. sqlSyntax
5. In IIS, create a new application and associate it to the same Application Pool as your existing QLM License Server.
6. Start the QLM application, go to Sites
7. Select your existing License Server
8. Click Copy and enter a new name of the copy
9. Update the URL to point to the new license server
10. Click on Upgrade Database Schema.
11. If you are using our eCommerce integration, do the following:
 1. Go to the Manage Keys tab
 2. Click on the Commerce Providers item in the toolbar
 3. Select the eCommerce provider you are using
 4. Modify the Dll field and replace the existing value with: QlmWebService.dll
 5. Ensure that the eCommerce Provider that you are using is enabled.

QLM Portal Upgrade

- On your web server, rename the QlmPortal folder to QlmPortal_old
- Copy %Public%\Documents\Quick License Manager\DeployToServer\QlmPortal to your web server in the same location as the previous QlmPortal folder
- Edit the web.config file in the new QlmPortal and update the following settings to match the values in the previous web.config file:
 - connectionStrings
 - communicationEncryptionKey
 - adminEncryptionKey
 - webServiceUrl
 - sqlSyntax

QlmCustomerSite Upgrade

- Note that QlmAspLicenseSite has been renamed to QlmCustomerSite
- On your web server, rename the QlmCustomerSite folder to QlmCustomerSite_old
- Copy %Public%\Documents\Quick License Manager\DeployToServer\QlmCustomerSite to your web server in the same location as the previous QlmCustomerSite folder
- Edit the web.config file in the new QlmCustomerSite and update the following settings to match the values in the previous web.config file:
 - connectionStrings
 - communicationEncryptionKey

- adminEncryptionKey
- webServiceUrl
- sqlSyntax
- Update the IIS Application to point to the QlmCustomerSite folder instead of the QlmAspLicenseSite folder.

To upgrade your source code to QLM v16:

- If you are upgrading from QLMv9 or earlier and if you are using the QLM License Wizard, be it the .NET Control or the standalone executable, you will need to customize the look & feel of the control and regenerate the settings xml file. Note that as of QLM v9, only one settings file is required. The UI Settings xml file is no longer required since all settings are stored in a single file.
- You may want to upgrade your LicenseValidator class to the new version. The new version contains additional code to optionally perform server side validation. It also supports a seamless reactivation process for subscriptions.
- If you have implemented floating licences, it is recommended that you review the new QLM Enterprise sample and follow the same approach as the new sample.

IMPORTANT -

QLM Engine Version

If you are upgrading from QLM v4 or earlier and you have issued license keys with QLM Engine version 4.0.00, 3.0.00, 2.4.11 or 2.4.07, we no longer support these keys. You must upgrade your customers to use keys created with QLM Engine Version 5.0.00.

Overview

Quick License Manager (QLM) is the ideal tool for software vendors who need to add licensing support to their products.

QLM uses several measures to defend the integrity of your software product. These include:

Tracking the date of first installation:

The date on which your software was first run on the user's machine is permanently recorded by QLM. Even if the user were to uninstall the product then re-install it at a later date, the original trial period would remain in effect.

Monitoring the system date for tampering:

If the user sets the system date back manually to defeat the evaluation period, QLM detects the tampering and disables the license.

Creating computer-bound keys:

A license key created under QLM can be bound to a single computer. Attempts to use the same key on other computers will not succeed.

Safeguarding the licensing DLL:

An easy-to-use mechanism allows you to verify at run-time that the code in QLM's licensing library has not been surreptitiously modified.

Using asymmetric encryption:

License keys created under QLM are encoded with an asymmetric encryption algorithm based on public and private keys.

Overview

QuickLicenseManager is the ideal tool for software vendors who need to quickly add license key support to their products. Both evaluation and permanent license keys can be generated.

Adding licensing support to your application is a 3 step process:

- Define your products.
- Generate license keys.
- Modify your application to capture and validate a license key.

The simplest way to get started with QLM is to launch the **Getting Started Wizard** by clicking on the **Get Started** tab then selecting **Guide Me**

For a quick overview of QLM, take a look at our online demos below.

Define products

The first step in creating license keys for your products is to define each product. Click on the **Define Products** tab and then click on **New**.

Enter the Product Name, the Major and Minor version of the product. The Product ID and the GUID fields are automatically generated. You will need these values when you use the API to validate license keys.

For more details about defining products, refer to the [Define Products](#) section in the User Interface Guide.

Generate license keys

Once you have defined your products, you can generate license keys for each product. There are 2 types of license keys: permanent, and evaluation keys. Evaluation keys can specify the duration of the evaluation, an expiry date or both. In case both an expiry date and a duration period are specified, the key will expire when any of the 2 criteria are met.

To generate license keys using **QLM Express**, click on the **Generate Keys** tab. Select a Product, the number of keys to generate, the type of keys to generate and then click on the **Generate** button. **Quick License Manager** typically generates unique license keys, however, under some circumstances it may be possible that a key value is duplicated. **Quick License Manager Express** does not store generated keys. It is up to the user to track the generated keys if needed. For more details about creating license keys with QLM Express, refer to the [Create Licenses](#) section in the User Interface Guide.

To generate license keys using **QLM Pro** or **Enterprise**, click on the **Manage Keys** tab, then click on the *Create* button. Select a Product, the number of keys to generate, the type of keys to generate and then click on the *OK* button. For more details about creating license keys, refer to the [Create Licenses](#) section in the User Interface Guide.

Modify your application

To capture and validate a license key in your code, use the [Protect your application](#) to generate the source code required to add licensing to your application.

Online Demos



[QLM Pro Demo](#)



[QLM Express Demo](#)



[Online Activation Demo](#)



[FastSpring \(eCommerce provider\) Integration Demo](#)

For more online demos, refer to our web site or run the Get Started Wizard in the QLM Console.

License key types

Quick License Manager supports 5 different types of license keys:

- **Evaluation or Subscription License Keys:** Evaluation keys can specify the duration of the evaluation, an expiry date or both. In case both an expiry date and a duration are specified, the license will be flagged as expired when the duration period is exceeded or the expiration date has passed.
- **Not Computer Bound or Generic:** Generic keys are not bound to a specific computer. The advantage of this type of key is that you can pre-generate a large set of keys and just distribute them to any customer. The disadvantage is that any user who gets access to a key is able to use that key on any computer.
- **Bound to Computer Name:** Computer Name bound keys are keys that are bound to a specific computer by encrypting the name of the computer in the license key. Computer Name bound keys can be generated from the Quick License Manager user interface or programmatically using the Quick License Manager API (CreateLicenseKeyEx). The advantage of Computer Name bound keys is that a key cannot be resued on other computers (unless they have the same name).
- **User Defined:** User defined keys are computer bound keys. The computer identifier that is used to uniquely identify a computer is user defined. For example, if you want to create a computer bound key that is bound to the serial number of the hard disk, you would need to do the following:

Create your own function to get the serial number of the hard disk.

To generate a key manually using the Quick License Manager user interface, you would need to generate the hard disk serial number in your application, display it to the user and ask the user to e-mail it to you. You would then enter this value in the Quick License Manager user interface and generate the key.

To generate a key programmatically, typically from a License Server as shown in the provided sample, you would generate the hard disk serial number in your code and invoke your License Server providing the serial number. The License Server would then use the CreateLicenseKeyEx API to generate and return a key bound to the serial number of the hard disk.

To validate a key in your code, you would generate the hard disk serial number and provide that value to the ValidateLicenseEx API.

- **Activation Key:** An activation key is a generic key that does not enable your software. It just allows a user to request a computer bound key. This type of key is typically used for online software registration or activation. This is a typical scenario:

A customer buys your software online. You automatically generate an activation key and send it to the customer.

The customer installs your software and enters the activation key. Your software connects over the internet to your License Server providing the information required to create a computer bound key. The License Server creates the key and returns it to your application.

Your application then validates the returned computer bound key and enables the software.

License key length

QLM packs several pieces of information in the license key. This information can be extracted at runtime to determine the type of license, the expiry date, the enabled features and so on. Depending on what information is packed in the license key, the length of the license key will vary.

Base Key Length	26 characters
Trial Key - Duration based	+3 characters
Trial Key - Expiry date based	+5 characters
With Features	+8 characters
With embedded seats (multiple activations)	+4 characters

Features

As of QLM 4.0, you can embed up to 32 features in a license key. Features are divided into 4 sets with 8 features per set.

To define features, click on Define Products, select a product and add your features.

You can generate license keys with embedded features in several ways:

- In the QLM console, click on Generate Keys, select the product and features to enable and then click on Generate.
- Using the QLM CreateLicenseKeyEx4 API (IsLicense50.dll)
- Using the QLM .NET API CreateActivationKey (QlmLicenseLib.dll)
- Using the QLM .NET API CreateActivationKeyWithExpiryDate (QlmLicenseLib.dll)
- Using the QLM .NET API CreateOrder (QlmLicenseLib.dll)

To enable support for 32 features, you need to select the QLM Engine version 4.0.00 or higher. To verify if a feature is enabled in your code, use the IsFeatureEnabled API. The sample located in:

%Public%\Documents\Quick License Manager\Samples\QLMExpress\DotNet\C#\QlmExpressSample demonstrates how to verify if a feature is enabled in your code.

Redistributables

The binaries that need to be included with a QLM protected application are listed in the table below.

Application Type	QLM Binaries to distribute
Windows Forms .NET 2.0 or higher	redistrib*.net2.0\QlmLicenseLib.dll redistrib*.net2.0\QlmControls.dll redistrib*.net2.0\QlmLicenseWizard.exe or redistrib*.net4.0\QlmLicenseLib.dll redistrib*.net4.0\QlmControls.dll redistrib*.net4.0\QlmLicenseWizard.exe
WPF .NET 4.0 or higher	redistrib*.net4.0\QlmLicenseLib.dll redistrib*.net4.0\QlmWpfControls.dll
ASP.NET 2.0 or higher	redistrib*.net2.0\QlmLicenseLib.dll or redistrib*.net4.0\QlmLicenseLib.dll
Office 2003 or higher	redistrib*.net2.0\QlmLicenseLib.dll redistrib*.net2.0\QlmLicenseWizard.exe or redistrib*.net4.0\QlmLicenseLib.dll redistrib*.net4.0\QlmLicenseWizard.exe
Outlook Add-in	redistrib*.net2.0\QlmLicenseLib.dll redistrib*.net2.0\QlmLicenseWizard.exe or redistrib*.net4.0\QlmLicenseLib.dll redistrib*.net4.0\QlmLicenseWizard.exe
C++	redistrib*.net2.0\QlmLicenseLib.dll redistrib*.net2.0\QlmLicenseWizard.exe or redistrib*.net4.0\QlmLicenseLib.dll redistrib*.net4.0\QlmLicenseWizard.exe
VB6	redistrib*.net2.0\QlmLicenseLib.dll redistrib*.net2.0\QlmLicenseWizard.exe or redistrib*.net4.0\QlmLicenseLib.dll redistrib*.net4.0\QlmLicenseWizard.exe

What is the difference between "QlmLicenseLib.dll" and "QlmLicenseLibNotEmbedded\QlmLicenseLib.dll"

QlmLicenseLib.dll includes all the QLM Express, Professional and Enterprise functionality (excluding user interface components). In addition, QlmLicenseLib.dll acts as a wrapper for IsLicense50.dll which is a C++ DLL that contains our core licensing engine. QlmLicenseLib.dll depends on IsLicense50.dll. IsLicense50.dll comes in 2 versions: one version of 32 bit systems and another version for 64 bit systems. When your application is running as a 64 bit app, we need to load the 64 bit version of IsLicense50.dll and similarly for 32 bit apps. To simplify deployment of the QLM binaries, QlmLicenseLib.dll includes the two versions of the IsLicense50.dll as resources. At runtime, the correct version of IsLicense50.dll is automatically extracted, temporarily stored on disk then loaded from disk. If it cannot be stored on disk due to security limitations, it is loaded directly from memory.

QlmLicenseLibNotEmbedded\QlmLicenseLib.dll is a version of QLMLicenseLib.dll that does not embed IsLicense50.dll. It is included for QLM 7 or before customers who used this deployment method in earlier versions of QLM. Unless advised by our support team, you should not need to use this version if you are using QLM 8 or later.

Protect your software

QLM protects against breaking the licensing in the following ways:

- QLM stores the date the software was first ran. If the user uninstalls the software and reinstalls it, the evaluation period continues from its previous state. This means users cannot just uninstall and reinstall the software to reinitialize the trial period.
- QLM detects if a user sets the date back and disables the license in this event.
- QLM allows you to define computer bound keys so that a license key cannot be reused on another computer.
- QLM provides a mechanism to detect if the QLM DLL was tampered.
- QLM license keys are protected with an encryption that each customer defines, per product. This means that other QLM customers cannot decrypt your license keys.

Protect against tampering of the QLM DLLs

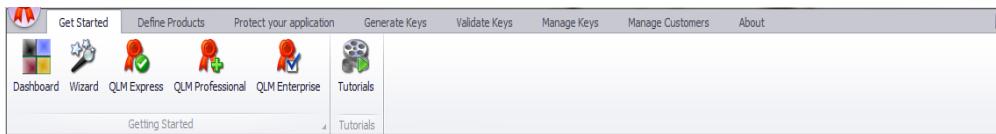
To protect against a user modifying the IsLicense50.dll or replacing this DLL, the IsLicense50.dll is digitally signed with a Microsoft Authenticode Certificate. The QLM .NET API automatically validates the signature of the DLLs prior to loading it.

All other QLM DLLs are .NET assemblies and are digitally signed as well. Any attempt at modifying these DLLs will result in a failure to load the DLL.

User Interface Reference

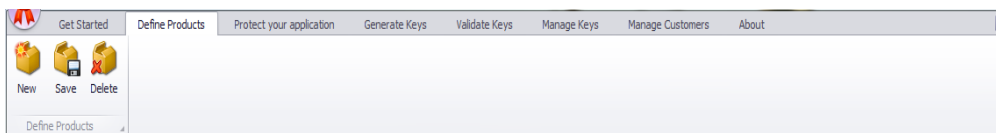
The QLM Console is a comprehensive command center for operating Quick License Manager. Across the top of the Console is a menu bar with commands for accessing each of eight main areas. These are introduced briefly here, then covered fully in the sections that follow.

Get Started



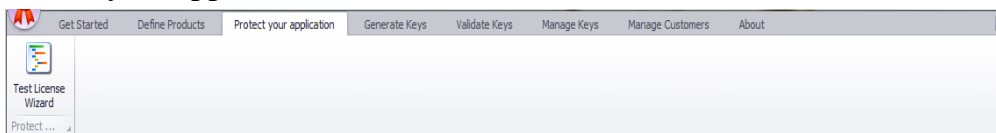
This area is home to the Dashboard, where you'll begin each Console session, and to the QLM Getting Started Wizard, an optional tool for helping you set up the licensing for one of your products efficiently. The wizard will recommend one of QLM Express, QLM Professional and QLM Enterprise as best meeting the licensing requirements of a particular product. It will also provide links to relevant informational and tutorial resources.

Define Products



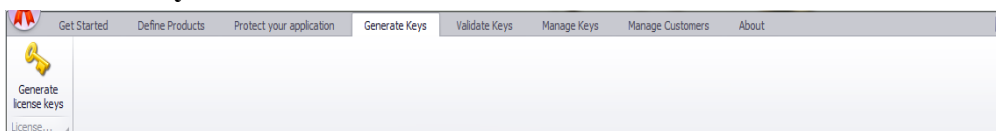
Before you can use QLM to manage the licensing of a product, you must add the product to QLM using the tools in this area. Once a product has been defined, you can begin configuring the keys and licensing options that will govern user access to the product's features.

Protect your application



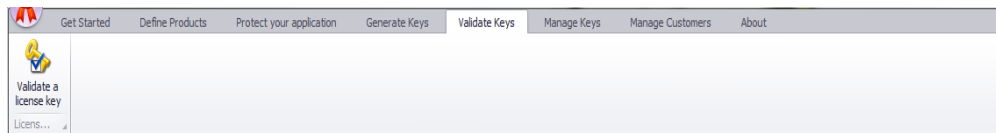
This area contains the Code Generator Wizard, which will generate code in any of several languages for adding license validation to your application. The wizard also lets you customize the built-in license registration form, if you choose to use it, for better integration with your product. To see how your configuration choices for the license registration form will look when it goes live, click the Test License Wizard button in the ribbon.

Generate keys



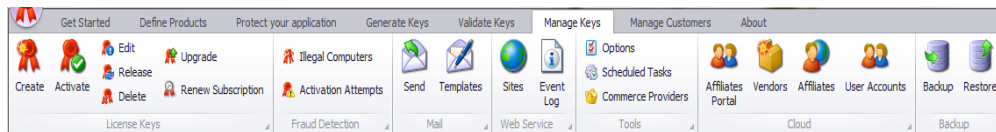
This area's purpose is to generate license keys for your products interactively (rather than in code). Several kinds of keys are supported, including the activation keys that let the user obtain permanent computer-bound or computer-independent keys. Such properties of the key as its expiry terms, and the particular set of features it will enable, can be defined with the controls here. Keys generated from this area are neither saved nor managed by QLM. To generate keys that are published to the database, create keys via the Manage Keys section.

Validate keys



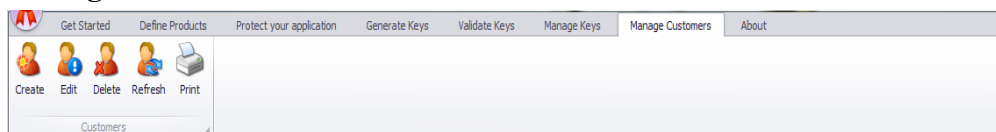
This area lets you check the validity of an existing key with respect to a particular product. If the key is valid, the set of features that it enables, the number of licenses it embeds, and its other properties are displayed.

Manage keys



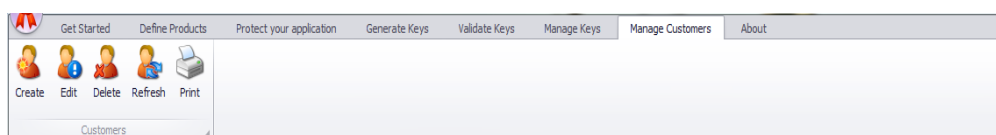
With its many controls, this area provides access to more than 20 license management functions in seven control groups: License Keys, Fraud Detection, Mail, License Server, Tools, Cloud and Backup. The tools and reports available here make Manage keys a particularly 'key' area of the Console's user interface.

Manage customers



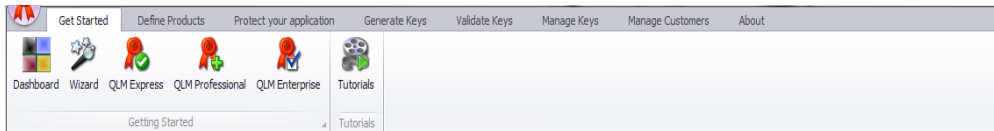
The customer records in your QLM database are the focus here. Controls are provided to create, edit, print and delete records, and to refresh synchronization with the database.

About



This area displays your licensing information for this copy of QLM. If you have not yet activated your license, you can do so here, using the same QLM license activation form provided as a programming component to QLM customers. Another control in this area checks for program updates; this too demonstrates the use of a QLM capability available to customers.

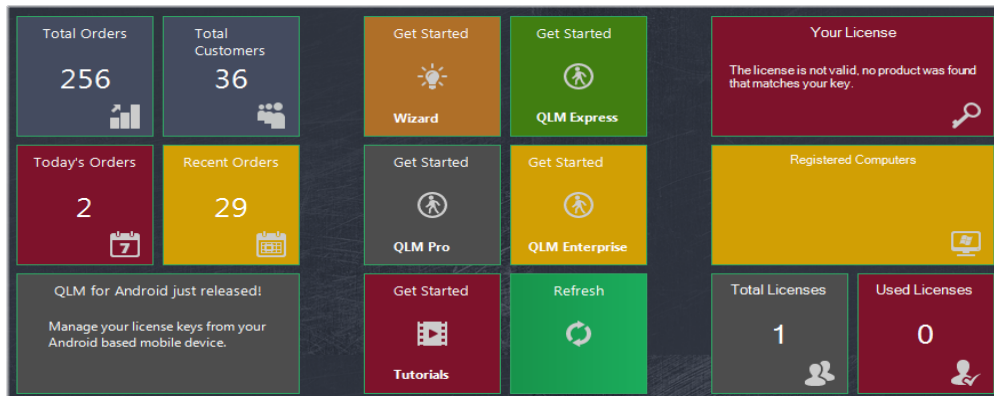
Get Started



The first of the QLM Console's eight major functional areas is accessed with Get Started at the left-hand end of the Console's main ribbon bar. Two Console features with which you will quickly become familiar are the Dashboard, which is the starting point of each session, and the Getting Started Wizard, an optional but handy tool for helping you set up product licensing under QLM. Both of these are found in the Getting Started control group, along with information pages detailing each of the three levels of QLM: Express, Professional and Enterprise. The single button in the Tutorials control group provides a link to online information resources with detailed information on selected topics.

THE DASHBOARD

The Metro-style tiles that make up the Dashboard are organized by default into three columns. A row in any column contains either one full-width or two half-width tiles. (If the window height is sufficiently reduced, doubled columns or tripled columns may be seen.) The organization of the Dashboard can be customized by dragging individual tiles from one column to another, or into an empty location to create a new column. As you drag a tile to locations in other columns, the tiles already there move aside as necessary to indicate where the new arrival will lodge. The default layout of Dashboard tiles in three columns.



System Information

The four smaller tiles in the left-hand column of the default layout show vital statistics from the QLM database, with counts of customers, total orders, today's orders, and recent orders. (See Manage Keys ? Options to review or adjust the meaning of 'recent'.) The bottom tile displays news items about QLM. Click an item to view the full story in your web browser.

Quick License Manager Information

The tiles in the center column provide information about the latest QLM version as well as News Releases about QLM. The Refresh tile resynchronizes the Dashboard display with the QLM database.

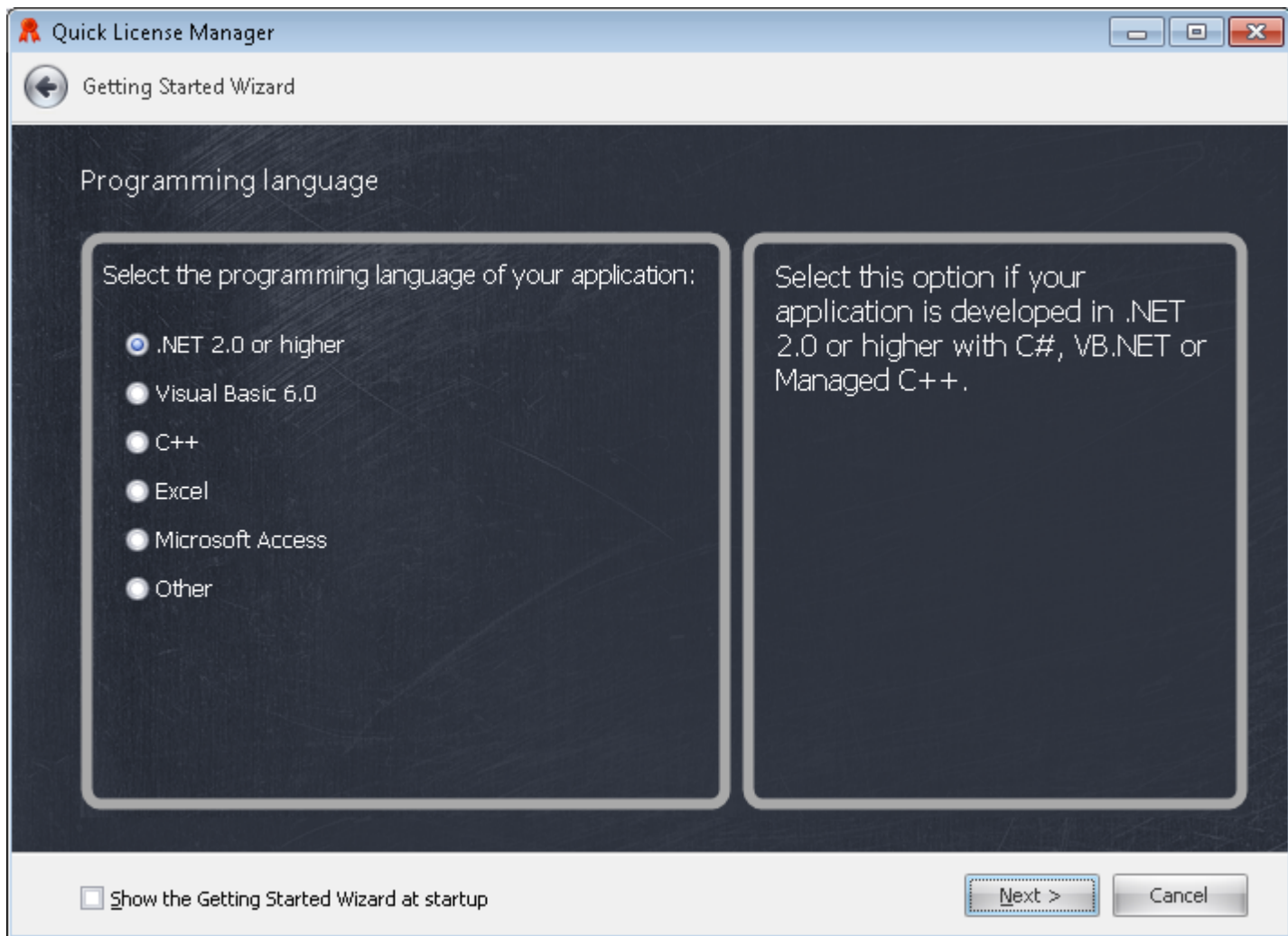
License Information

The tiles in the right-hand column of the default Dashboard layout show information about your QLM license. The upper box gives a verbal status report regarding the license. The one beneath it lists the computers on which you are registered to run QLM. In the bottom row are displayed the number of licenses you have available, and the number of those already in use.

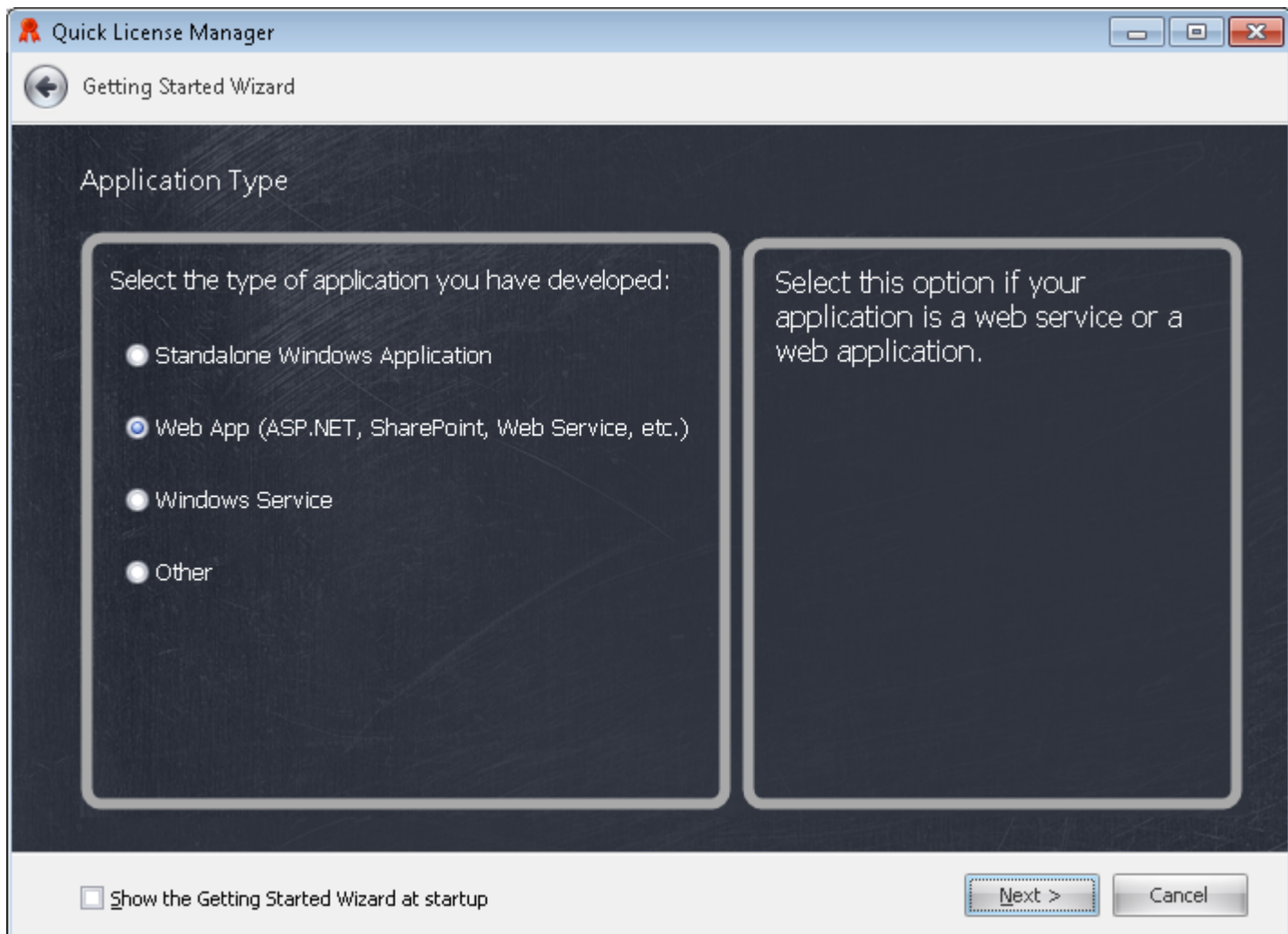
GET STARTED WIZARD

The Getting Started Wizard helps you find the best path into QLM for a particular licensing scenario by leading you through a questionnaire to determine three requirements:

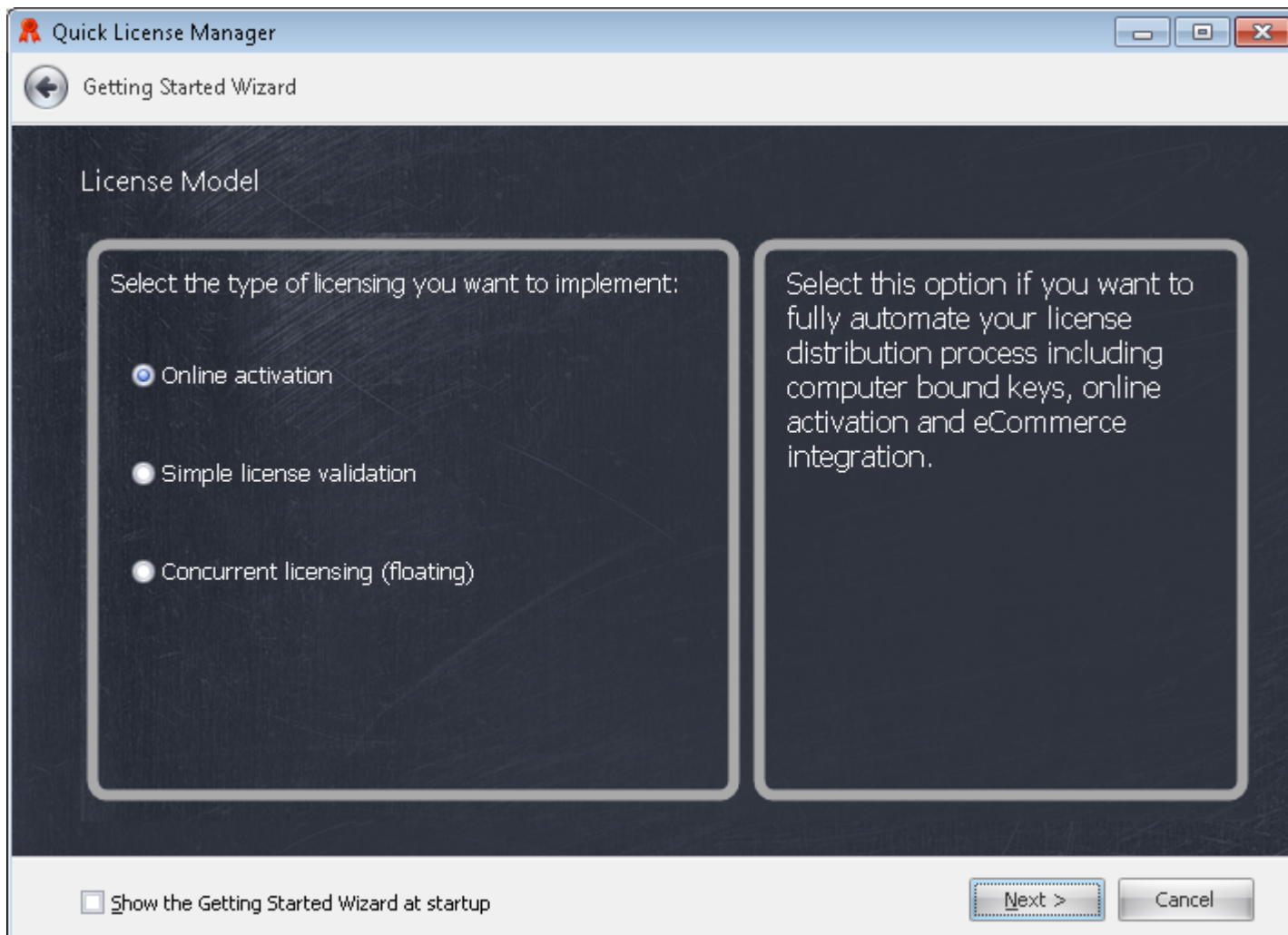
Programming language: .NET 2.0 and higher languages are directly supported by QLM, as are Visual Basic 6.0, C++, Excel and Access. Other languages can also make use of the QLM API if they are capable of interfacing with .NET assemblies, COM objects (ActiveX), or native DLLs.



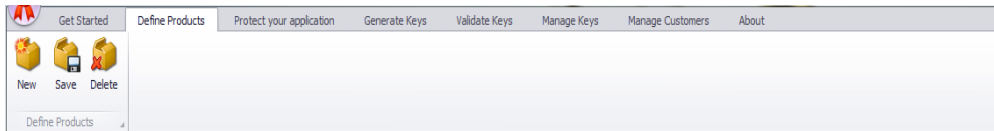
Application type: The wizard offers “Standalone Windows application”, “Web App”, and “Windows Service” as configured choices. If your application does not fall into one of these categories (“Other”), contact us for advice specific to your situation.



License model: Your answer to this question determines the edition of QLM that you will need for your application. For instance, if you select “Simple license validation” as your license model, the wizard will recommend QLM Express as best meeting your needs. The wizard then provides a page of links to relevant tutorials and code samples that will help get you started without wasted effort.



Define Products



A QLM license key is associated with a particular product as identified by its name and version in the Define Products tab.

To add a product, click on the Add button and enter the product information.

The full set of properties is divided amongst four tabs, *Product Information*, *Latest Version*, *Keys* and *Vendor*.

PRODUCT INFORMATION

Product name: Enter the base name of your product without versioning.

Major version, Minor version: This pair of numbers, both of which can have at most two digits, specify a particular generation and release of the product for licensing. Taken together, the name and the two version fields constitute a unique handle by which this product will be identified in QLM.

Encryption key: Enter an encryption key to encrypt the license key. An encryption key is like a password. A license key can only be decrypted with this password. The ! character is not allowed in the encryption key. Encryption keys are no longer used as of QLM 5. A PKI algorithm is now employed for encrypting the license key. Each time you define a product in QLM, a key pair is created for it. Typically a different key pair is used for each product. You will use the private key in the pair to generate license keys. The public key is used to decrypt license keys. It must be included in your code and set before licenses can be validated.

Product ID: This field is automatically generated when the database record for the product is stored, and remains permanently associated with it from then on.

Release Date: The Release Date is used in conjunction with the Maintenance Plan feature to determine upgrade entitlement. Provided the Maintenance Plan Renewal Date is greater than the Release Date, the customer is entitled to the upgrade.

GUID: This identifier is automatically generated for your project by QLM. It is used as a key to the evaluation information for the product in Windows Registry under:

HKEY_CLASSES_ROOT\CLSID<GUID>. The New button adjoining the field generates a new GUID. Calls to the DefineProduct API in your application will need to be updated if you do this.

Features: Up to four feature sets with up to eight licensable features in each can be defined for your product. The licenses for the product can be configured on creation to enable one or more of these sets, thus making their individual features available to the user who holds the license.

The information about the features you have so far defined for the product is found on the Features list. Each line displays the set number (0 to 3), a numeric ID, and a name provided by you. The numeric values used as IDs are those corresponding to the individual bits of a flag byte: 1, 2, 4, and so on up to 128. To manage the Features list, three buttons are provided: Add, to define a new feature; Edit, to change the assignable properties of the selected feature – its name and the set that it belongs to; and Remove to discard the feature altogether.

LATEST VERSION

The latest version tab has three fields that let you implement a “Check for Updates” feature in your

software.

Latest Version: Enter in this field the latest version of your product. The format of this value is up to you. The availability of a particular update will be determined in your running software by retrieving this value and comparing it with the current version.

URL to latest version: Enter a URL to the latest version of your software for the QLM updating framework to use.

Notes about latest version: The notes you enter in this text box can be retrieved for display to the user along with other information about the update.

KEYS

QLM uses asymmetric encryption to encrypt license keys. The algorithm requires that a pair of encryption keys, one public and one private, be predefined. QLM generates such a pair for you automatically when your product is created. By default the same key-pair is used across all versions of the same product.

Public key: You will need this value in order to validate a license in your program code.

Private key: This value is used for generating licenses. For the security of your licensing process, it is recommended not to generate licenses in the protected software itself, but to use an external mechanism for that purpose.

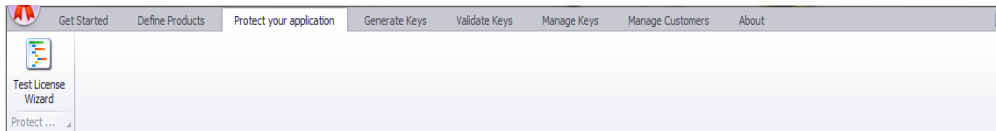
New: Click this button to generate a new key pair at any time. Remember, however, that your software must be updated to use the new public key value.

Unmask: Check this box to view the actual text of your private key rather a string of asterisks.

VENDOR

The Cloud Edition of QLM allows you to associate a product with a particular vendor, by selecting the vendor from the dropdown list and entering the vendor's own unique ID for the product.

Protect your application



QLM provides several approaches to protect your application. The Protect your application wizard generates a helper class to include in your application as well as 2 configuration files that specify the properties of the QLM License Wizard Control. The QLM License Wizard Control is a license registration form to capture a license key and activate it.

The **Protect your application** tab in the QLM Management Console walks you through the steps to protect your application. On the first step of the wizard, select the product you want to protect, the QLM License Server and your programming language.

On step 2 of the wizard, select the Look & Feel options of the QLM License Wizard.

;

On Step 3 of the wizard, select the licensing options. These options reflect the properties of the QlmLicense object as described in the API Reference section in the online help.

Finally, select a location where QLM will save the helper class and the xml files with your customizations and click Save.

If your application is a C# or a VB.NET application, QLM can automatically update your Visual Studio project with the required references and helper files.

PROTECTING .NET APPLICATIONS

For Windows Forms .NET applications, QLM provides 3 .NET Controls that you can easily drop in your application. These controls are forms that allow the end-user to enter a license key and activate it.

The 3 .NET controls are:

- QLM Express: QlmExpressLicenseValidationControl (QlmControls.dll)
- QLM Pro/Enterprise: QlmWebBasicActivationControl (QlmControls.dll)
- QLM Pro/Enterprise: QlmLicenseWizardCtrl (QlmControlLicenseWizard.dll).

Both QLM Pro/enterprise controls offer very similar functionality. The main difference between these 2 controls is that the QlmLicenseWizardCtrl uses a wizard based graphical user interface. In addition, the QlmLicenseWizardCtrl can read its properties from 2 external configuration file. These configuration files are generated by the **Protect your application** wizard.

The QLM License Wizard Control is also available as a standalone executable that can run alongside your application.

When you install QLM on your system, a Quick License Manager tab is added to your Visual Studio toolbox that contains the QLM .NET Controls. If for any reason the Quick License Manager tab was not added to your Visual Studio toolbox, you can attempt to recreate this section by clicking on the Refresh button under Options / Enable Visual Studio Integration. Note that the QLM tab is not added to the Visual Studio Express edition as this edition does not support programmatic additions to its toolbox.

For more details about the QLM .NET Controls, refer to the API Reference.

For WPF applications, you can host the QLM Windows Forms Controls in WPF as

described in this [article](#).

If you are developing a web based **ASP.NET** application, the above .NET controls cannot be used. A sample program is available that shows how to capture and activate a license key. The sample is located here:

%Public%\Documents\Quick License

Manager\Samples\qlmpro\Windows\DotNet\Basic\C#\vs2008\AspDotNetSample

In addition to the .NET Controls that are required for capturing and activating a license key, the **Protect your application** wizard generates a helper class that you need to add to your application. The helper class has a method called **ValidateLicenseAtStartup**. You should call this method when your application is launched. For more details about this method, refer to the API reference section.

PROTECTING NON .NET APPLICATIONS

For non .NET applications, you have 2 options to capture and activate a license: (a) you can use the standalone *QlmLicenseWizard.exe* application to capture and activate a license or (b) you can create your own license registration form. The former is clearly the simplest approach.

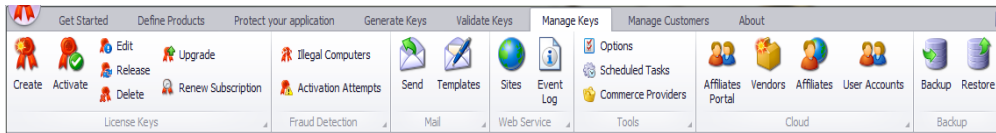
For VB or C++ applications, the C++ applications, the **Protect your application** wizard generates a helper class that you need to add to your application. The helper class has a method called *ValidateLicenAtStartup*. You should call this method when your application is launched. If the call to **ValidateLicenseAtStartup** fails or returns that activation is needed, you should then invoke the *QlmLicenseWizard.exe* as follows:

```
QlmLicenseWizard.exe /settings "<path>\settings.xml" /uiSettings  
"<path>\uiSettings.xml"
```

where both xml files referenced above are generated by the **Protect your application** wizard.

When the *QlmLicenseWizard.exe* application exits, you should once again call **ValdiateLicenseAtStartup** and confirm that the license is valid. If it is not, you either exit your application or launch the *QlmLicenseWizard.exe* application again.

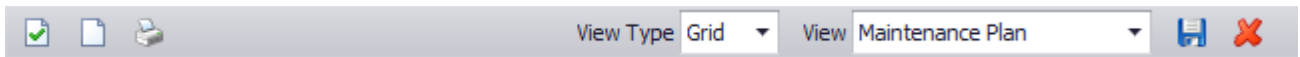
Manage keys



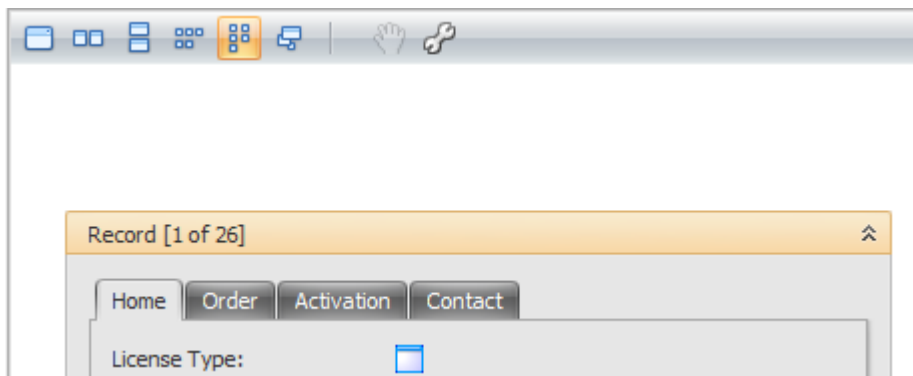
The Manage Keys area in the QLM Console provides a task-oriented interface to the QLM database located on your web server. Communication between the Manage Keys section and the database occurs via the QLM License Server which must be configured from the Sites button.

The main window permits interactive configuration of the selection, ordering and grouping of result columns. A query builder across the top of the display lets you access data according to whatever combinations of criteria you specify.

On the right, a column of preset queries provides answers to often-asked questions such as, "How many licenses have we sold today?", "How many activation licenses were obtained yesterday for our products?", and "Which customers on maintenance licenses are up for renewal in the near future?".



Items are shown in a grid or card view. To switch between the grid view and the card view, click on the View Type dropdown box in the status bar and select the view of your choice. Views can be customized and saved. To save a view, click on the Save icon in the status bar. To customize the fields displayed in the card view, click on the wrench button on the card view toolbar.



LICENSE KEYS

The buttons in this control group allows you to create or manage licenses. Note that all operations described in this section can also be automated via the QLM API. Some operations are also available via regular http requests that can be invoked directly from an eCommerce provider.

CREATE



This button opens the Create License dialog window for creating activation keys interactively and publishing them to the QLM database. Providing such a key to a customer at the time of sale begins a licensing transaction that ends when the customer successful enters the activation key into your installed application and a computer bound license key is generated and sent back to the customer.

ACTIVATE



Activates an activation key and generates a computer bound license key. Use this button to activate one or more keys interactively for testing purposes or as other circumstances may require. Ordinarily, activation should be performed by your software while running on the customer's machine. To activate a key, you need to specify a unique computer identifier for the customer's computer. The Computer Name field is optional but recommended.

EDIT



This button brings up the Edit License Information dialog window, which provides controls for editing the data associated with a license.

RELEASE



Customers may sometimes require a transfer of their license key to another computer. The Release button clears the existing association between this license and a computer; once this has been done, the license can be reactivated on the new system. QLM maintains a history of all released licenses that can be queried by means of the **Search Released Licenses** item in the Search dropdown at the top right of the area header.



DELETE

This button deletes, after confirmation, one or more licenses selected in the data browser. Deleting a license key removes all records associated with the key.



UPGRADE

Upgrade your customer to a new version of your product, an additional set of features, or a new version of the QLM Engine. The upgrade process generates a new activation key, which you must convey to the customer for use. The previous activation key is archived.



RENEW SUBSCRIPTION

If you sell your software as a subscription, you can renew subscriptions without sending your customers a new license key by moving back the subscription expiry date using the Renew Subscription date window. Customers simply reactivate their existing license keys to renew their subscriptions. When the subscription associated with a license is extended, a computer-bound key is generated with the revised expiry date. The customer receives the new key by activating the license.

CREATE

The Create button creates an activation key. Activation keys can also be created via the QLM API and more typically via QLM's integration with eCommerce providers. For more information about how to automatically create activation keys during the purchasing process, read the eCommerce Providers section in the QLM Professional help.

- **Product:** The product associated with the license key.
- **Number of keys:** The number of license keys to generate.
- **Number of activations per key:** The number of activations that this key will allow. QLM manages the number of activated licenses and prevents the user from activating more than the number of purchased licenses. For example, if a customer purchases 5 copies of your software, rather than sending the customer 5 license keys, you create a single license key that can be activated on 5 different computers.
- **Floating seats:** To implement Floating Licenses (requires QLM Enterprise), specify the number of floating seats associated with this activation key. You can alternatively use this field to control the number of instances of your application that can run in a Terminal Server session. An instance is uniquely identified by a Terminal Server session for a specific user. To enable this feature, set the LimitTerminalServerInstances property of the QlmLicense object to True in your code.
- **Customer E-mail:** Associate the key to a customer.
- **Features:** Check any feature to enable in this license key (if you have not defined any features for your product, this field will be empty).
- **Affiliate:** Select from this dropdown list the affiliate that sold the license, if applicable.
- **Generic License:** When this option is checked, the license key to be generated upon activation will be generic – not bound to any particular computer.
- **Maintenance Plan:** When selected, license keys become version-agnostic. A customer running an older version of your product will be able to activate his license key with the latest version of your product. Use this option if you have offered your customers free upgrades to all future versions or if the customer has purchased a maintenance plan. For more details, read the Maintenance Plan section in the QLM Professional help.
- **Expiry Criteria:** If you would like the license to be time-limited, check this box to enable its subordinate controls.
- **Engine Version:** If you have customers using an older version of QLM, select the appropriate version of the QLM engine from this dropdown list on the Advanced tab. It is highly recommended to use QLM Engine version 5.0 or higher.
- **User data:** Use this text area on the Advanced tab to record any notes or text-encoded data to associate with this license.
- **Comments:** Use this text area on the Comments tab to record any notes or text-encoded data to associate with this license.

Finally, the Save Defaults button allows you to store default values for all the fields above. Subsequently launching the Create dialog will automatically load the saved default values.

ACTIVATE

The Activate button manually activates the selected license. Note that activation typically occurs from within your application via the QLM API. Manual activation through the QLM Console is required in a situation where the end-user cannot activate a license due to the lack of an internet connection. This is typically referred to as Manual Activation v/s Online Activation.

- **Activation Key:** The key to activate (this field is read-only).
- **Customer E-mail:** The customer's email address from an existing customer record.
- **Computer Identifier:** A unique identifier for the customer's computer. This can be the computer name, the MAC address, the hard disk serial number or any other identifier of your choice.
- **Computer Name:** The name of the computer. Setting this field is not required but is recommended for easy identification of the customer's system.
- **Product:** The product associated with the license key.
- **Engine Version:** The version of the QLM Engine. If you have customers using an older version of QLM, select the appropriate version of the QLM engine.
- **Affiliate:** The version of the QLM Engine. If you have customers using an older version of QLM, select the appropriate version of the QLM engine.
- **User Data:** Associate any data to the license key

Export and Print

You can print or export license keys or customer information. When exporting, QLM can export to the following file format: PDF, HTML, RTF, XLS, CSV, TXT or to an image file.

Below are the steps required to export or print data:

- Click on the Manage Keys or Manage Customers tab, depending on what you want to print or export.
- Run a search to display the data you would like to print or export.
- On the status bar, locate the **Print** button and click it.
- In the Preview window that is displayed, click the **Print** button or the **Export Document** button.
- You can also customize the look and feel of the data by clicking on the **Customize** button.

FRAUD DETECTION

QLM can track illegal computers that connect to the License Server and logs information about these computers in the database. An illegal computer is defined as a computer that has a valid license key but whose license key is (a) not in the database or (b) in the database but registered for another computer. This situation can occur if a user with a valid license key requests that his license be released from computer A claiming to have uninstalled your program from computer A. If the user subsequently attempts to connect to the License Server via computer A, QLM detects this computer as an illegal computer and logs it in the database.



ILLEGAL COMPUTERS

This button displays a list of all the detected illegal computers. QLM does not prevent all illegal computer from running your application. Once an illegal computer is detected, it is up to you to decide the course of action to take by contacting the customer and enquiring about the situation.



ACTIVATION ATTEMPTS

Click the Activation attempts button to view a list of failed activation attempts. A failed activation is typically due to multiple attempts to activate the same license key on different computers. Though failed activation attempts may indicate an intention to illegally activate a license on a particular computer, they can also be regarded as the sign of a user who needs additional licenses for your software.

MAIL

QLM provides a tool to send personalized emails to your customers. Note that QLM sends emails via your Outlook client.

SEND

To send a personalized email to a set of customers:

- Perform a search that returns a set of data.
- Select the items that you want to send an email to or click on the Select All button.
- Click on the Mail/Send button.
- Select the Outlook profile to use.
- Select the email template to use.
- Select the E-mail account to use.
- The **To** field should already contains the email of the selected customers.
- The additional **To** field allows you to type additional email addresses. Note that if your email message uses variables, these variables will not be expanded when sending emails to the additional recipients.
- Enter the Subject of the email.
- Enter the body of the email.
- Click on the Send button.



To create a personalized message, you can use variables within the **Subject** and the **Body** of the message. Any of the visible columns in License Management tab may be used as variables.

For example, your message could read:

Hi %FullName%,
Thank you for your recent purchase of
%ProductName%
%MajorVersion%.%MinorVersion%.
Your license key is: %ActivationKey%.
Regards,
Tom

TEMPLATES

Templates allow you to create common email content that can be readily used when sending manual or automated emails to your customers. Automated emails can be configured from the Scheduled Tasks option.



Aggregated Emails

Typically, QLM sends an email to the customer associated with the license along with any other email you specify in the CC or BCC fields. In some cases however, you may want to send an *internal* email to yourself or to someone within your company instead of the customer. To do so, you simply uncheck the "Send To Customer" check box. When sending an *internal* email, rather than receiving one email for each affected record, you could configure QLM to send you an email that aggregates results from several records. This is accomplished by adding special tags in the email body to identify the repeatable section.

There are two ways to configure an email to aggregates multiple records into a single email:

Standard text format

If your email is formatted as plain text, you can delimit the repeatable section with the following tags:

- *****qlm_content_start*****
- *****qlm_content_end*****

qlm_content_start identifies the beginning of the repeatable section whereas qlm_content_end identifies the end of the repeatable section.

Table format

If your email is formatted as plain text with a repeatable section formatted as an html table, you can delimit the repeatable section with the following tags:

- *****qlm_table*****

qlm_table identifies the beginning of the repeatable section. Your table must start right after this tag. There is no need to specify a tag to delimit the end.

LICENSE SERVER

The options in this control group pertain to the QLM License Server, which acts as an interface between a client system and the QLM database. Although you will typically have a single License Server, the QLM Console can manage a list of License Server profiles, each with its own server address along with data such as the type of database engine on the server, the encryption keys for communicating with the License Server and so on. Until you set up your own License Server, and during your QLM trial period, you can use the Default License Server provided by Soraco. Note that the Default License Server does not allow you to upload your own products and can only manage keys for the built-in Demo product that ships with the trial version of QLM.

SITES

Click this button to open the License Server Settings dialog window, which lets you create, edit and select your connection to the QLM License Server.

- **URL:** The URL to the QLM License Server is typically of this form:
`http://yourdomain/yourvirtualdirectory/qlmservice.asmx`
- **Authentication Method:** Anonymous | Forms Authentication | Windows Authentication. In most cases, you will want to configure the web service for Anonymous Authentication.
- **Database:** Select whether the database you installed on the server uses Microsoft Access or Microsoft SQL Server.
- **Database Schema:** When a new version of QLM is released, the QLM database schema may require updating. This button verifies and updates your DB schema as required.
- **Path to products file:** The products you define in QLM are initially stored locally on your computer. Once you are satisfied with your products definition, your products need to be uploaded to the QLM database. QLM automatically detects if your products are out of sync with the server and prompts you to either upload or download products from the server. If you want to force an update, click on the appropriate button in this section. Use this feature with caution as you may inadvertently overwrite the QLM database with the wrong products or vice versa.
- **Communication/Admin Encryption Key:** Communication between a client and the QLM License Server is protected via an encryption mechanism that prevents hackers from directly calling your License Server. This is critical due to the fact that the License Server is typically configured to allow anonymous connections.



- **Test:** Once you have configured all fields, click on the Test button to validate that you can properly connect to the QLM License Server.

EVENT LOG

Errors detected in the QLM License Server are stored in the QLM database. Use this option to view these errors. At times, you may want to increase the verbosity of the messages that the QLM License Server logs. This can be accomplished by updating the **loggingLevel** setting in the web.config file of the QLM License Server. The highest logging level is 15. The recommended value is 3.



If you increase the **loggingLevel** to 15 to diagnose a specific issue, remember to set it back to 3 to avoid bloating of your QLM database.

QLM Server Properties

Following is a list of all the server properties that control the behaviour of the QLM License Server. These properties can be set from the QLM Management Console under Manage Keys / Sites / Server Properties.

Category	Name	Description
Options		
	allowMultipleProductVersionsOnSameSystem	By default, if a user tries to activate a new version of a given product on the same computer, QLM will reuse the existing activation and overwrite previous data. If you need to allow a customer to install different versions of your product on the same system and activate each separately, set this property to true.
	countRevokedTrials	Determine whether revoked trials should be taken into account when evaluating the numberOfTrialLicensesAllowedPerClient.
	dbMaxRecords	Controls the maximum number of license keys that can be created.
	defaultCommerceProvider	The default ecommerce provider. This ecommerce provider is used if the is_vendor argument is not specified in the URL.
	defaultLicenseModelWhenDurationSet	When creating a license key with an expiry duration, if the LicenseModel value is not set, the system will automatically set the LicenseModel to this value. Possible values are: permanent trial subscription.
	defaultLicenseModelWhenExpiryDateSet	When creating a license key with an expiry date, if the LicenseModel value is not set, the system will automatically set the LicenseModel to this value. Possible values are: permanent trial subscription.

defaultQlmVersion	Default version of the QLM Engine. This version is used if the <code>is_qlmversion</code> argument is not specified in the URL.
enforceMaxTrialsWhenActivating	Controls whether QLM limits the number of trial keys per system. The number of trial keys allowed per system is controlled with the <code>numberOfTrialLicensesAllowedPerClient</code> property.
historyTableLogRelease	When a license is deactivated, QLM automatically logs a copy of the license record prior to deactivation in the QLM history table. You can turn off this feature by setting the <code>historyTableLogRelease</code> property to false. If you use QLM Pro Cloud based floating licenses, it is highly recommended to set the <code>historyTableLogRelease</code> to false.
historyTableLogUpgrade	When a license is upgraded, QLM automatically logs a copy of the license record prior to the upgrade in the QLM history table. You can turn off this feature by setting the <code>historyTableLogUpgrade</code> property to false.
licenseKeyFormatGroupSize	When a license key is generated, a separator (-) will be inserted every <code>n</code> characters. This property specifies the size of <code>n</code> .
licenseKeyFormatMaxGroupSize	When a license key is generated, a separator (-) will be inserted every <code>n</code> characters as specified by the <code>licenseKeyFormatGroupSize</code> property. The last group in the license key may not be equal to <code>n</code> . This property specifies the maximum number of characters in the last group.

loggingLevel	Controls the QLM logging level. Set this value to 15 for the highest possible logging level.
maintenancePlanGracePeriod	An expired maintenance plan cannot be renewed. This property allows for a grace period after expiry where the maintenance plan can still be renewed.
maintenancePlanPeriodInDays	Default maintenance plan period when the maintenance plan is enabled.
maxActivationsEnforcedOnVMs Only	Enforce the maxActivationsPerSystem property only if the ComputerType is VM.
maxActivationsPerDay	Limits the number of activations that can be performed in one day.
maxActivationsPerSystem	Limit the number of activations on the same system. This is useful in the context of virtual machine to prevent users from cloning VMs and activating the same license on multiple virtual machines.
maxNewKeysPerDay	Limits the number of new activation keys that can be created in one day.
maxReleaseCount	The maximum number of times an end-user can release a license.
maxReleasePerClient	When counting the number of released licenses for a given activation key, count only the ones associated to a specific client. By default, QLM counts all the released licenses for a given activation regardless of the client system.
maxReleasePeriodInDays	When counting the number of released licenses, only count the ones that have been released in the past

	<p>"maxReleasePeriodInDays" days. For example, if you want to allow a user to release a license twice per month, set maxReleasePeriodInDays to 30 and maxReleaseCount to 2.</p>
numberOfTrialLicensesAllowedPerClient	<p>Limits the number of trial licenses that can be activated by a client. A client is uniquely identified by a ComputerID and a ComputerName.</p>
releaseLicenseRequiresAuthentication	<p>By default, releasing (de-activating) a license via the ReleaseLicenseHttp method requires the user to authenticate. This property allows you to relax this requirement and invoke ReleaseLicenseHttp without authentication.</p>
releaseLicenseUseAdminEncryptionKey	<p>By default, releasing (de-activating) a license programmatically does not require the AdminEncryptionKey property to be set. This property enforces the need to set the AdminEncryptionKey property prior to calling ReleaseLicense.</p>
renewMaintenancePlanWhenSubscriptionRenewed	<p>When renewing a subscription via the RenewSubscriptionHttp method, automatically set the Maintenance Renewal Date to the same value as the Subscription Expiry Date.</p>
restrictManagementApiByIP	<p>Calling the QLM Management API requires knowledge of the AdminEncryptionKey. For additional security, you can limit access to the QLM Management API to a set of IP addresses (semi comma separated). Note that the local IP 127.0.0.1 is implicitly allowed to call any QLM Management API. Use this feature with care because you could lock yourself out of QLM. If</p>

	<p>you do lock yourself out, you must delete the value of the <code>restrictManagementApiByIP</code> property in the <code>ServerProperties</code> table in the QLM database. Alternatively, you can install QLM on the server and update the <code>restrictManagementApiByIP</code> property using the QLM Management Console. This is guaranteed to work because all local requests bypass this validation mechanism.</p>
<code>sqlSyntax</code>	<p>If you are using a MS-SQL Server, set this property to sql. If you are using MS-Access, set this property to ms-access</p>
<code>subscriptionGracePeriod</code>	<p>An expired subscription cannot be renewed. This property allows for a grace period after expiry where the subscription can still be renewed.</p>
<code>trialDuration</code>	<p>Specify the duration of the trial period used when calling the <code>CreateComputerBoundTrialKey</code> API.</p>
<code>updateExistingUserInfo</code>	<p>When creating a new license for a user via an ecommerce provider, if the user already exists, update the user information (except the email address), based on the new order.</p>
<code>useDurationToSetExpiryDate</code>	<p>When creating an Activation Key with an expiry duration, the QLM License Server can convert the specified duration to a specific expiry date based on the current date. The <code>SubscriptionExpiryDate</code> field will be set to this expiry date.</p>
<code>useDurationToSetExpiryDateWhenActivating</code>	<p>When activating a duration based license, the QLM License Server can convert the specified duration</p>

to a specific expiry date based on the time of activation. The SubscriptionExpiryDate field will be set to this expiry date.

The QLM License Server methods are protected against being intercepted and replayed at a later time. All methods are time stamped. When the server detects that the time stamp does not match the server's time, the method fails. This property allows you to specify the maximum allowed time discrepancy between the end user system's and the server. Time stamps are time zone independent. The default value is 600 seconds or 10 minutes.

webMethodMaximumDelay

Paypal

ignoreCustomArgument

Ignore data in the paypal **custom** field

ignoreItemNumberArgument

Ignore data in the paypal **item number** field

paypalLoggingLevel

Controls the paypal logging level. Set this value to 15 for the highest possible logging level.

paypalUrl

URL to the paypal IPN processor. By default this value is configure to connect to the paypal sandbox. Once you are ready to go live, you need to change this URL to point to the real paypal IPN process:
<https://ipnpb.paypal.com/cgi-bin/webscr>

qlmVersion

Version of the QLM Engine. Should be set to 5.0.00 unless you want to generate license keys compatible with QLM v4.0 and earlier.

qlmWebServiceUrl

URL to the QLM License Server,

such as
http://server/qlm/qlmservice.asmx.

revokeLicenseOnRefund When a paypal order is refunded, automatically revoke the license.

revokeLicenseWhenSubscription Canceled When a paypal subscription is canceled, automatically revoke the license.

templateFile Once an order is processed successfully, QLM sends an email to the customer with the license key information. The email template is name is defined by this property. The file should be located in the same folder as the License Server.

vendorCompanyEmail Email address of the vendor, i.e. your email address.

vendorCompanyName Company name of the vendor, i.e. your company name.

Regional Settings

dateFormat Format of a date (no time) sent by a client via an http method call. The default value is yyyy-MM-dd.

eCommerceProviderDateFormat Format of a date (no time) sent by an eCommerce provider via an http method call. The default value is yyyy-MM-dd HH:mm:ss.

sqlDateFieldFormat Date format use in SQL queries. The default value is YYYY-MM-DD.

sqlDateFormat Date format use in SQL queries. The default value is dd/MM/yyyy HH:mm:ss.

sqlDateOnlyFormat Date format use in SQL queries. The default value is YYYY-MM-DD.

sqlDateTimeFormat	Date time format used in SQL queries. The default value is: yyyy-MM-dd HH:mm:ss.
sqlDateValueFormat	Date format use in SQL queries. The default value is yyyy-MM-dd.

Security Settings

enableCreateActivationKey	Allows users to create an activation key via the API.
enableCreateComputerBoundTrial	Allows users to create a computer bound key via the CreateComputerBoundTrialKey API.
enableCreateOrder	Allows users to create an order via the API.
enableGetActivationKey	Allows users to create an activation key via the the http call GetActivationKey.
enableGetLatestVersionHttp	Allows users to get information about the latest version via an http call.
enableRegisterLicense	Allows users to register a QLM portal license.
enableReleaseKeyHttp	Allows users to release (de-activate) a license via an http call.
enableRenewSubscriptionHttp	Allows users to renew a subscription via an http call.
enableUpgradeDatabase	Allows users to perform a DB upgrade.
enableUpgradeLicense	Allows user to register their license.
enableUploadAffiliates	Allows users to upload affiliates.
enableUploadECommerceProviders	Allows users to upload ecommerce providers.

	enableUploadProducts	Allows users to upload products.
	enableUploadUserAccounts	Allows users to upload user accounts.
SMTP Settings		
	smtpEnableSSL	Enables SMTP over SSL.
	smtpFrom	Email address of the sender.
	smtpFromDisplayName	Display name of the sender.
	smtpPassword	Password associated to the SMTP user.
	smtpPort	Port used by the SMTP server.
	smtpServer	IP or DNS name of the mail server.
	smtpUser	Name of the SMTP user.
CustomerSite		
	allowGenericEmailProviders	Determines if generic email providers (such as hotmail, gmail) can request license keys. You can customize the list of generic email providers in the web.config file.
	isPhoneRequired	Determines if the phone field is mandatory in the Trial Registration Form.
	isFullNameRequired	Determines if the full name field is mandatory in the Trial Registration Form.
	isEmailRequired	Determines if the email field is mandatory in the Trial Registration Form.
	isCompanyRequired	Determines if the company field is mandatory in the Trial Registration Form.
	isCountryRequired	Determines if the country field is mandatory in the Trial Registration

	Form.
maxRegistrationsPerUser	The maximum number of registrations allowed per user (email). The default is 1.
preventMultipleRegistrationsPerDomain	Prevent a user from requesting a trial license key if another user from the same domain has already requested a trial. This property works in conjunction with the genericEmailProviders property that allows you to configure generic email providers such as gmail.com and hotmail.com.
preventMultipleRegistrationsPerMajorVersion	Prevent a user from requesting multiple license keys for the same product major version. The default value is True.
preventMultipleRegistrationsPerMinorVersion	Prevent a user from requesting multiple license keys for the same product minor version. The default value is True.
preventMultipleRegistrationsPerProduct	Prevent a user from requesting multiple license keys for the same product. The default value is True.
preventRegistrationsIfCustomer	Prevent a user from requesting a trial license key if another user from the same domain has already purchased this product. This property does not impact customers that use generic Email Providers.
registerButtonDoneUrl	The URL that the page redirects to in the Trial Registration Form when the user clicks the Done button after registration is completed.

TOOLS

The buttons in this control group pertain to miscellaneous options such as QLM console configuration, Affiliates Portal configuration, Dashboard settings and eCommerce Provider options.



OPTIONS

- **Path to searches files** : Searches configured in the right hand panel are stored in a file called queries.xml. You can configure the path of this file. This is typically used if you want to share searches with different people on your team and store searches on a shared path (UNC) on your LAN.
- **Display options**: When you create a query on the fly by selecting the "field", "operator" and "value" in the toolbar over the data set grid, QLM can display the details of your query in the status bar as you build it.
- **Recent Orders**: Allows you to configure the horizon of the recent orders displayed in the QLM dashboard.
- **Upcoming Maintenance Renewals**: Allows you to configure the horizon of the upcoming maintenance renewals displayed in the QLM dashboard.
- **URL to Affiliates Portal**: Allows you to configure the URL to the Affiliates Portal.
- **QLM Agent**: Allows you to configure whether the QLM Agent automatically starts when you login. The QLM Agent is responsible for running the QLM Scheduled Tasks.

SCHEDULED TASKS

Click the Scheduled Tasks button to open a dialog window where you can define operations to be carried out automatically. The scheduled tasks that QLM can execute include emailing notifications to your customers, and displaying alerts when the QLM database is updated. QLM installs with several scheduled tasks predefined and ready for use. One task is designed to handle the email notifications sent to maintenance plan customers; the others are of the alert type. You can use the supplied tasks as-is or customize them to suit your needs. You can also define additional scheduled tasks if required.



Note that as you transition from a trial version of QLM to a purchased version with your own License Server, you should modify the scheduled tasks to point to your

own License Server instead of the Default/Demo License Server.

COMMERCE PROVIDERS



The Commerce Providers button opens the Edit Commerce Providers dialog widow, which lets you quickly configure integration between your instance of the QLM License Server and any of the major online commerce providers. The typical settings that you will need to modify are the User / Password. For more details, review the eCommerce Provider section under QLM Professional.

QLM PORTAL

The QLM Portal is an add-on to QLM. It provides a web based interface for managing license keys. The QLM Portal exposes the most common operations an administrator or a reseller requires to manage their customers' license keys. It requires user authentication and uses role based access control. The QLM Portal is a perfect tool for your resellers or affiliates to manage their own customers while keeping you, the vendor, in total control. You can specify how many license keys a reseller can generate, which products they have access to and what operations they can perform. The QLM Portal requires QLM Pro or Enterprise v7.2 or above and a SQL Server Database.

Through a role based access model, you control the privileges of user or group of users. The license key limits that can be set are:

- Maximum number of trial keys per system.
- Maximum number of permanent keys per system.
- Maximum total keys.
- Maximum activations per key.

In addition, you can control which of the following operations a user is allowed to perform:

- Creating new keys.
- Activating keys.
- Releasing keys.
- Deleting keys.
- Creating customers.
- Deleting customers.
- Exporting keys.
- Setting the Expiry Duration of a new key.
- Setting the Expiry Date of a new key.
- Setting the Maintenance Plan option.
- Setting the Generic license option.



QLM PORTAL

The QLM Portal button provides a shortcut to access the QLM Portal. To configure the URL of the QLM Portal, click on Tools / Options button.

USER ACCOUNTS

User accounts are used to limit access to the QLM Portal.

A user account can be associated to a User Group.

When a user logs to the QLM Portal with an account associated to a User Group, that user will only see license keys associated to that User Group.





USER GROUPS

QLM allows you to define User Groups that share common privileges. For example, if you sell your application through a reseller, you would create a User Group for the reseller and then create a user account for each employee of the reseller that needs to manage license keys.

MANAGE CUSTOMERS

The Manage Customers area of the QLM Console provides five functions for managing customers' records in the QLM database.

CREATE



Create a new customer in the QLM database. Customers are uniquely identified by their email address. If you define 2 customers with the same email address, the records of the 2 customers will be merged.



EDIT

Edit the information of the selected customer.



DELETE

Delete the record of the selected customer.



REFRESH

Refresh the list of customers.

Database Backup and Restore

QLM can schedule and perform backups of your database to your local computer. No special software is required on the server side to perform the backup or the restore.

Below are the steps required to create a backup job:

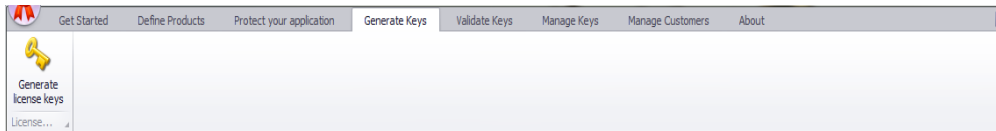
- Click on the Backup tab.
- Click on the **Create Backup** button.
 - Select the Enable checkbox.
- Type the name of the backup job.
- If you have multiple License Servers, select the License Server profile to use. Otherwise, select the Default profile.
- Specify how often the backup job will run.
- If you have configured a Disaster Recovery site, specify whether you would like the backup to be automatically restored to the DR site by checking the Auto Restore checkbox.
- If you have configured a Disaster Recovery site (DR), and the DR site is active, the backup will by default backup the active site, be it the primary site or the DR site. If you would like to always backup the primary site, whether it is active or passive, check the Always Backup Primary Site option. Note that if the primary site is not active, the Auto Restore feature is automatically disabled.

To restore a backup job, follow the steps outlined below:

- Click on the Restore tab.
- Expand the backup job and select the snapshot to restore.
- Select the tables to restore. Note that data in the target tables will be deleted.
- Click on the Restore button or Restore to DR site button.

QLM provides basic feedback as to whether the primary site and the DR sites are synchronized by comparing the number of records in the LicenseKeys table on both sites. The first time you click on the Backup tab, comparison between both sites is automatically triggered. You can request to compare two sites at any time by clicking on the Compare Now or Compare All buttons.

Generate Keys



The **Generate Keys** tab of the QLM Console lets you configure and create a license key – or a list of license keys – for any of your products.

Note that license keys generated from this tab are not stored in the **QLM** database. To create license keys that can be used for online activation, refer to the *Create* option under the **Manage Keys** tab.

Select your product on the *Product Name* dropdown, then continue with the other controls in the Options box as follows:

Number of embedded licenses: Enter here the number of licenses to embed in each generated key. For example, if you enter “2” the user holding one of these keys will be permitted to activate it on two different computers.

Number of license keys to generate: Enter the number of license keys to generate when you click the *Generate license keys* button. This options is designed for working with online payment systems that issue the next available key from a prepared list when your software is purchased. Note that licenses distributed in this way do not benefit from QLM’s database integration.

Engine version: Specify the version of the QLM Engine to use. If your application has shipped with an earlier version of QLM, select the version of the QLM Engine that your application is using. Note that the latest version of the QLM Engine is 5.0.00.

Permanent License: Check this option if you want the generated license keys to remain in effect permanently once issued.

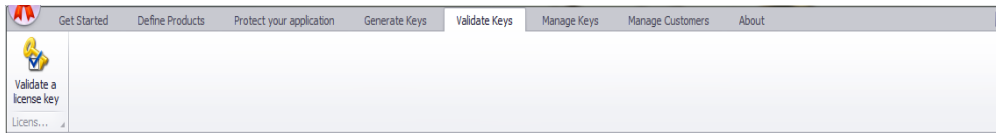
Expiry Criteria: Check this option for trial or subscription based licenses that should expire within a certain number of days of installation, or upon a predetermined date.

Features: Check the boxes next to the features that you want this license to enable. The list of available features can be configured in the *Define Products* tab.

License Type: This dropdown selects which of the four available QLM license types will be generated. The “Bound to Computer Name” and “User Defined” types both request additional information in the *Computer Identifier* field.

When you have completed your license settings, click the *Generate license keys* button in the ribbon strip. The key, or set of keys, you have requested appears in the *License Keys* box. Click the *copy* button to the right of the box to transfer the information to the Windows Clipboard. The final destination – a customer email, a configuration page on-line, or elsewhere – is up to you.

Validate Keys



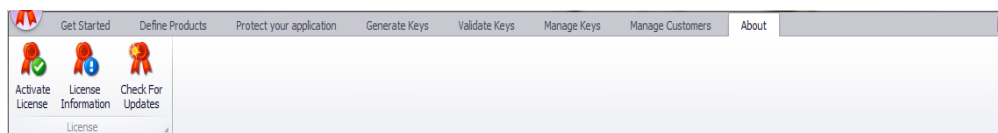
The **Validate Keys** area of the QLM Console allows you to interactively test a key to determine whether it is valid.

To validate To validate a license key, paste it into the *License Key* text field and click **Validate a license key**.

If the license is valid, the information contained in the key is decrypted and displayed in the form.

IMPORTANT: For computer bound license keys, you need to enter the *Computer Identifier* field prior to validating the license. QLM cannot validate a computer bound license key without the *Computer Identifier*.

About



This tab displays licensing information about your QLM copy.

ACTIVATE LICENSE

When you purchase QLM, you receive an email with a license key that typically starts with the letter 'A'. To activate your purchased copy of QLM:

- Click the License Wizard button
- Click Activate your license
- Click Activate Online
- Enter your Activation key then click Activate

The activation process connects over the internet to the Soraco QLM License Server and activates your license. If you connect to the internet via a proxy server, click on the Proxy Settings button to configure your proxy server.

DEACTIVATE A LICENSE

If you wish to transfer your license to another computer, you can deactivate your license then activate it on a different system. To deactivate your license:

- Click the License Wizard button
- Click Deactivate your license
- Click Deactivate

Note that you are allowed 4 transfers per year.

CHECK FOR UPDATES

To check if you are running the latest version or to upgrade to the latest version:

- Click the License Wizard button
- The welcome page displays the version you are running and the latest version. If a more recent version is available, click the "Update to the latest version" link
- Review the Release notes
- Click Download
- When the Download completes, click Install

QLM Express License Validation .NET Control

The QLM Express License Validation .NET Control is a Windows Forms based .NET control that you can easily add to your Windows Forms application to capture and validate a license key.

The control can be found in the Visual Studio toolbox, in the Quick License Manager section.

To integrate QLM Express with your application, you also need the automatically generated helper class.

This class can be generated from the [Protect your application](#) wizard.

Following is a list of all the properties that can be set on the QLM Express License Validation .NET Control.

Name	Description
QlmCloseButtonVisible	Show or hide to Close button
QlmComputerID	Set the computer ID to use when activating the license. This property should be typically set programmatically at runtime.
QlmEncryptionKey	Set the encryption key. This property is only required when using QLM engine version 4.0 and earlier.
QlmEvaluationHaveKeyRadioButtonText	Only applies if the QlmEvaluationVisible property is set to true. Set the text in the radio button when the user has a license key.
QlmEvaluationLicenseKey	Only applies if the QlmEvaluationVisible property is set to true. Set the evaluation key to use when the user selects to evaluate the software and does not have a license key.
QlmEvaluationTrialChecked	Only applies if the QlmEvaluationVisible property is set to true. Checks the evaluation option by default.
QlmEvaluationTrialHelpText	Only applies if the QlmEvaluationVisible property is set to true. Set the text to display under the evaluation radio button.
QlmEvaluationTrialRadioButtonText	Only applies if the QlmEvaluationVisible property is set to true. Set the text to display on the evaluation radio button.
QlmEvaluationVisible	Enables the evaluation option. The evaluation option displays two radio buttons. One radio button allows the user to enter a license key and activate the license while the other radio button allows the user to evaluate the software by using an embedded evaluation license key.
QlmFormBackColor	Set the starting Background Color of the form to

	produce a gradient effect.
QlmFormBackColor2	Set the ending Background Color of the form to produce a gradient effect.
QlmGUID	Set the GUID associated to your product. The GUID can be found on the Define Product page in the QLM Console.
QlmHeaderBackColor	Set the Background Color of the header pane.
QlmLicenseStatus	Get the status of the license after it has been validated. This is a read-only property.
QlmLicenseType	Set the license type. The license type can be: ComputerName, UserDefined or Generic.
QlmLogoFont	Set the font to use in the logo text.
QlmLogoImage	Set the image to use for the logo.
QlmLogoText	Set the text to use for the logo.
QlmMajorVersion	Set the Major Version associated to your product. The Major Version can be found on the Define Products page in the QLM Console.
QlmMinorVersion	Set the Minor Version associated to your product. The Minor Version can be found on the Define Products page in the QLM Console.
QlmProductID	Set the Product ID Version associated to your product. The Product ID can be found on the Define Products page in the QLM Console.
QlmProductName	Set the Product Name associated to your product.
QlmPublicKey	Set the Public Key associated to your product. The Public Key Version can be found on the Define Products page (Keys tab) in the QLM Console.
QlmStoreKeysLocation	By default, QLM stores the license keys in a hidden file on the end user system. You can also select to store the license keys in the registry by setting this property.
QlmValidateCertificate	The QLM DLLs are digitally signed by a trusted certificate authority. In order to ensure that hackers

do not replace the QLM DLLs by dummy ones,
QLM can validate that the DLLs are properly
signed.

Distribute your application

The sections below describe the options to include the QLM required DLLs in your application. Select the most appropriate option based on the type of setup that you use to deploy your application.

Using the QLM .NET API from VB6 or unmanaged C++

If you have included a reference to QlmLicenseLib.dll in your VB6, unmanaged C++ or any other non .NET based application, your setup must perform the following operations:

32 bit applications

- **Generate a type library to be referenced by your code**
 - `%windir%\Microsoft.NET\Framework\v2.0.50727\regasm.exe /tlb "<fullpath>\QlmLicenseLib.dll"`
- **Register the QlmLicenseLib.dll as a COM object**
 - `%windir%\Microsoft.NET\Framework\v2.0.50727\regasm.exe /codebase "<fullpath>\QlmLicenseLib.dll"`

64 bit applications

- **Generate a type library to be referenced by your code**
 - `"%windir%\Microsoft.NET\Framework64\v2.0.50727\regasm.exe" /tlb "<fullpath>\QlmLicenseLib.dll"`
- **Register the QlmLicenseLib.dll as a COM object**
 - `"%windir%\Microsoft.NET\Framework64\v2.0.50727\regasm.exe" /codebase "<fullpath>\QlmLicenseLib.dll"`

Distribute your application using Merge Modules

InstallShield and other MSI based installation tools

If you are using Windows Installer to distribute your application, you can use the merge modules found in the Quick License Manager **redistrib\MergeModules**

folder. When including the merge module, set the destination folder of the merge module to be your application's installation folder.

Application Type	QLM Express - Merge Modules to include	QLM Professional or Enterprise - Merge Modules to include
Windows Forms .NET 2.0 or higher	Soraco Quick License Manager 5.0 (IsLicense50.msm) or (recommended) Soraco Quick License Manager 5.0 (IsLicense50.msm) Soraco Quick License Manager 5.0 for .NET 2.0 (QlmLicense.net2.msm) Soraco Quick License Manager User Controls 5.0 for .NET 2.0 (QlmControls.net2.msm)	Soraco Quick License Manager 5.0 (IsLicense50.msm) Soraco Quick License Manager 5.0 for .NET 2.0 (QlmLicense.net2.msm)
ASP.NET 2.0 or higher Excel 2003 or higher MS-Access 2003 or higher Outlook Add-in VB6	Soraco Quick License Manager 5.0 (IsLicense50.com.msm) or (recommended) Soraco Quick License Manager 5.0 for .NET 2.0 (QlmLicense.net2.emb.msm)	Soraco Quick License Manager 5.0 for .NET 2.0 (QlmLicense.net2.emb.msm)
C++	Soraco Quick License Manager 5.0 (IsLicense50.com.msm) & or (recommended) Soraco Quick License Manager 5.0 (IsLicense50.msm)/p> Soraco Quick License Manager 5.0 for .NET 2.0 (QlmLicense.net2.com.msm)	Soraco Quick License Manager 5.0 (IsLicense50.msm) Soraco Quick License Manager 5.0 for .NET 2.0 (QlmLicense.net2.com.msm)

Note that if your application does not target x64 bit platforms, i.e. your application runs as a 32 bit application on 64 bit machines, then you should include the IsLicense50_x86.msm instead of IsLicense50.msm.

&

The IsLicense50.msm merge module includes 2 files:

- FileName: em>IsLicense50.dll
- Destination Folder: [INSTALLDIR]
- ComponentCode: {CD707E5F-495E-48E5-8360-EAB26AFC186A}
- Shared: No
- COM Extract at Build or Self-Register: No
- Permanent: No
- Architecture: x86

- FileName: *IsLicense50.dll*
- Destination Folder: *[INSTALLDIR]*
- ComponentCode: {752E6A64-1248-46B5-87E5-8039A54F6288}
- Shared: *No*
- COM Extract at Build or Self-Register: *No*
- Permanent: *No*
- Architecture: *x64*

The QlmLicenseLib.net2.msm merge module include several files:

- FileName: *QlmLicenseLib.dll*
- Destination Folder: *[INSTALLDIR]*
- ComponentCode: {84A850A3-7CDE-4E23-B31E-58C4C716E54F}
- Shared: *No*
- COM Extract at Build or Self-Register: *No*
- Permanent: *No*
- FileName: *QlmLicenseLib.resources.dll for Spanish*
- Destination Folder: *[INSTALLDIR]es*
- ComponentCode: {560F660B-2649-45A9-BB53-D499DE808F34}
- Shared: *No*
- COM Extract at Build or Self-Register: *No*
- Permanent: *No*
- FileName: *QlmLicenseLib.resources.dll for French*
- Destination Folder: *[INSTALLDIR]fr*
- ComponentCode: {DC0FB90A-AE8A-4261-A82E-BC18B7DFB4C7}
- Shared: *No*
- COM Extract at Build or Self-Register: *No*
- Permanent: *No*
- FileName: *QlmLicenseLib.resources.dll for Italian*
- Destination Folder: *[INSTALLDIR]it*
- ComponentCode: {F040D81F-AC78-4CC9-9BD7-18B4F881F34F}
- Shared: *No*
- COM Extract at Build or Self-Register: *No*
- Permanent: *No*
- FileName: *QlmLicenseLib.resources.dll for German*
- Destination Folder: *[INSTALLDIR]de*
- ComponentCode: {7C02CB03-FB75-4B80-95C9-0F6421E578C1}
- Shared: *No*
- COM Extract at Build or Self-Register: *No*
- Permanent: *No*

Distribute your application using a Visual Studio Deployment Project

Visual Studio .Net Deployment Project

To add Quick License Manager to your Visual Studio .Net deployment project, follow the steps below:

- Create a deployment project
- Add your project's output in the File System Editor
- This should detect the Quick License Manager dependencies and include the following files:
 - IsLicense50.dll - If you are targeting x64 bit systems, you will need to conditionally install the proper IsLicense50.dll depending on the target platform.
 - The x64 bit version of IsLicense50.dll is located in the Redistrib\x64 folder.
 - QlmLicenceLib.dll
 - QlmControls.dll
 - *es\QlmLicenseLib.resources.dll*
 - *fr\QlmLicenseLib.resources.dll*
 - *it\QlmLicenseLib.resources.dll*
 - *de\QlmLicenseLib.resources.dll*

Application Type	QLM Express - Files to include	QLM Professional or Enterprise - Files to include
Windows Forms .NET 2.0 or higher	redistrib\x86\IsLicense50.dll redistrib\x64\IsLicense50.dll or (recommended) redistrib\x86\IsLicense50.dll redistrib\x64\IsLicense50.dll redistrib\.net2.0\QlmLicenseLib.dll redistrib\.net2.0\QlmControls.dll	redistrib\x86\IsLicense50.dll redistrib\x64\IsLicense50.dll redistrib\.net2.0\QlmLicenseLib.dll redistrib\.net2.0\QlmControls.dll
ASP.NET 2.0 or higher Excel 2003 or higher MS-Access 2003 or higher Outlook Add-in VB6	redistrib\x86\IsLicense50.dll redistrib\x64\IsLicense50.dll or (recommended) redistrib\.net2.0\QlmLicenseLibEmbed\QlmLicenseLib.dll redistrib\.net2.0\QlmControls.dll	redistrib\.net2.0\QlmLicenseLibEmbed\QlmLicenseLib.dll redistrib\.net2.0\QlmControls.dll
C++	redistrib\x86\IsLicense50.dll redistrib\x64\IsLicense50.dll or (recommended) redistrib\x86\IsLicense50.dll redistrib\x64\IsLicense50.dll redistrib\.net2.0\QlmLicenseLib.dll redistrib\.net2.0\QlmControls.dll	redistrib\x86\IsLicense50.dll redistrib\x64\IsLicense50.dll redistrib\.net2.0\QlmLicenseLib.dll redistrib\.net2.0\QlmControls.dll

Distribute the QLM DLLs with your application

Some or all of the following files must be included in the setup of your application. To determine which files to include in your setup, check the table later in this section. All QLM binaries should be installed in the same folder as your application.

When you include the IsLicense50.dll, you must create two folders, x86 and x64, then copy the corresponding IsLicense50.dll in each folder.

Note that all .NET DLLs come in two versions: a .NET 2.0 version and a .NET 4.0 version. You can find all QLM DLLs in the redistrib folder:

x86\IsLicense50.dll: not shared, to be installed in the same folder as your application.

x64\IsLicense50.dll: not shared, to be installed in the same folder as your application

QlmLicenseLib.dll: not shared, to be installed in the same folder as your application

QlmControls.dll: not shared, to be installed in the same folder as your application

QlmControlLicenseWizard.dll: not shared, to be installed in the same folder as your application

QlmLicenseWizard.exe: not shared, to be installed in the same folder as your application

To determine what DLLs to include in your application, refer to the table below.

Application Type	QLM Express - Files to include	QLM Professional or Enterprise - Files to include
Windows Forms .NET 2.0 or higher	redistrib\x86\IsLicense50.dll redistrib\x64\IsLicense50.dll or (recommended) redistrib\x86\IsLicense50.dll redistrib\x64\IsLicense50.dll redistrib\.net2.0\QlmLicenseLib.dll redistrib\.net2.0\QlmControls.dll redistrib\.net2.0\QlmControlLicenseWizard.dll	redistrib\x86\IsLicense50.dll redistrib\x64\IsLicense50.dll redistrib\.net2.0\QlmLicenseLib.dll redistrib\.net2.0\QlmControls.dll redistrib\.net2.0\QlmControlLicenseWizard.dll
ASP.NET 2.0 or higher Outlook Add-in	redistrib\x86\IsLicense50.dll redistrib\x64\IsLicense50.dll or (recommended) redistrib\.net2.0\QlmLicenseLibEmbed\QlmLicenseLib.dll redistrib\.net2.0\QlmLicenseWizard.exe	redistrib\.net2.0\QlmLicenseLibEmbed\QlmLicenseLib.dll redistrib\.net2.0\QlmLicenseWizard.exe
Excel 2003 or higher MS-Access 2003 or higher VB6 Delphi Any language that interfaces with COM/ActiveX	redistrib\x86\IsLicense50.dll redistrib\x64\IsLicense50.dll or (recommended) redistrib\.net2.0\QlmLicenseLib.dll redistrib\.net2.0\QlmLicenseWizard.exe	redistrib\.net2.0\QlmLicenseLib.dll redistrib\.net2.0\QlmLicenseWizard.exe
C++	redistrib\x86\IsLicense50.dll redistrib\x64\IsLicense50.dll or (recommended)	redistrib\x86\IsLicense50.dll redistrib\x64\IsLicense50.dll redistrib\.net2.0\QlmLicenseLib.dll

	redistrib\x86\IsLicense50.dll redistrib\x64\IsLicense50.dll redistrib\.net2.0\QlmLicenseLib.d ll redistrib\.net2.0\QlmControls.dll	ll redistrib\.net2.0\QlmControls.dll
--	--	---

Localization

The QLM .NET API calls may return messages relating the status of a license key validation. By default, all messages returned by the QLM .NET API are in English. The QLM .NET API also supports many languages such as Spanish, Italian and German messages. To configure your application to display the proper message depending on the language of your customer's system, you need to:

- Locate the localization folders in the Quick License Manager "redistrib\Localization" folder.
- When you deploy your application, copy these folders to the same location as the QlmLicenseLib.dll
- For example, if your customer's system is running Spanish OS, copying the "es" folder will result in the Spanish resources to be automatically used.

Alternatively, if you would like to force a specific language, you need to add the following call to your application prior to initializing any UI:

```
System.Threading.Thread.CurrentThread.CurrentUICulture = new  
System.Globalization.CultureInfo("es-ES");
```

Additionally, the QLM .NET Controls as well as the QLM License Wizard are localized. The QlmControl.resources.dll contains the localized resources for the QlmControls.dll whereas QlmLicenseWizard.resources.dll contains the localized resources for the QLM License Wizard.

64 bit Support

When distributing your application, depending on the QLM features that you are using, you may need to distribute the following DLLs with your application:

- IsLicense50.dll
- QlmLicenseLib.dll
- QlmControls.dll

IsLicense50.dll

is a C++ DLL. If you are targeting a 64 bit platform, you need to ship with your application the 64 bit version of this DLL. The 64 bit version of the DLL is found in the x64 folder. The *QlmLicenseLib.dll* and *QlmControls.dll* are .NET assemblies. The same version of these DLLs will work on 32bit or 64 bit platforms.

If your application is intended to run as 32 bit application on a x64 bit operating system, then you need to use the 32 bit version of IsLicense50.dll. If you are using the provided merge modules to integrate QLM into your setup, you should use the IsLicense50_x86.msm or IsLicense_x86.com.msm. These merge modules install the 32 bit version of the IsLicense50.dll regardless of the target platform.

BackwardCompatible

Set this property to true to allow validation of keys prior to the latest version of the QLM engine.

C++: VARIANT_BOOL BackwardCompatible

C#: bool BackwardCompatible

CreateLicenseKey

Creates a non-computer bound license key. If the ExpiryDate is NULL and the ExpiryDuration is -1, the license key is a permanent non-evaluation license key.

Prior to calling this function, you must call DefineProduct and set the PrivateKey property. Note that including the PrivateKey in your code is not recommended. Creation of license keys should not typically be done from within the application but rather from a server that the user does not have access to.

C++: `_bstr_t CreateLicenseKey (DATE ExpiryDate, int ExpiryDuration)`

C#: `string CreateLicenseKey (System.DateTime ExpiryDate, int ExpiryDuration)`

Parameters

ExpiryDate - The date when the license will expire. Use NULL if you do not want to specify an expiry date.

ExpiryDuration - The duration of the evaluation period in days. Use -1 if you do not want to specify a duration.

Return

A non-computer bound license key.

CreateLicenseKeyEx

Creates a computer bound license key.

Prior to calling this function, you must call DefineProduct and set the PrivateKey property. Note that including the PrivateKey in your code is not recommended. Creation of license keys should not typically be done from within the application but rather from a server that the user does not have access to.

C++: `_bstr_t CreateLicenseKeyEx (ELicenseType LicenseType, BSTR MachineID)`

C#: `string CreateLicenseKeyEx (ELicenseType LicenseType, string MachineID)`

Parameters

LicenseType - Specify the type of license to generate. See the definition of LicenseType below.

MachineID - A unique identifier for the machine. If you specify a *ComputerName* as the LicenseType, this argument must be the Computer Name. If you specify *User Defined* as the LicenseType, this argument can be anything you want. When validating the license key in your code, you will need to provide the same value to the ValidateLicenseEx function.

Return

A computer bound license key.

CreateLicenseKeyEx2

Creates a computer bound license key that has an expiry date and a number of licenses.

Prior to calling this function, you must call DefineProduct and set the PrivateKey property. Note that including the PrivateKey in your code is not recommended. Creation of license keys should not typically be done from within the application but rather from a server that the user does not have access to.

C++: `_bstr_t CreateLicenseKeyEx2 (DATE ExpiryDate, int ExpiryDuration, int NumberOfLicenses, ELicenseType LicenseType, BSTR MachineID)`

C#: `string CreateLicenseKeyEx2 (System.DateTime ExpiryDate, int ExpiryDuration, int NumberOfLicenses, ELicenseType LicenseType, string MachineID)`

Parameters

ExpiryDate - The date when the license will expire. Use NULL if you do not want to specify an expiry date.

ExpiryDuration - The duration of the evaluation period in days. Use -1 if you do not want to specify a duration.

NumberOfLicenses - The number of licenses associated with the key. Use 1 if you do not want to use single activation licensing.

LicenseType - Specify the type of license to generate. See the definition of LicenseType below.

MachineID - A unique identifier for the machine. If you specify a *ComputerName* as the LicenseType, this argument must be the Computer Name. If you specify *User Defined* as the LicenseType, this argument can be anything you want. When validating the license key in your code, you will need to provide the same value to the ValidateLicenseEx function. To create a key that is not computer bound, set this argument to NULL and set the LicenseType to Generic.

Return

A computer bound license key.

CreateLicenseKeyEx3

Creates a computer bound license key that has an expiry date, a number of licenses and a specific set of features that are enabled.

Prior to calling this function, you must call DefineProduct and set the PrivateKey property. Note that including the PrivateKey in your code is not recommended. Creation of license keys should not typically be done from within the application but rather from a server that the user does not have access to.

C++: `_bstr_t CreateLicenseKeyEx3 (DATE expiryDate, int expiryDuration, int numberOfLicenses, ELicenseType licenseType, BSTR machineID, in features)`

C#: `string CreateLicenseKeyEx3 (System.DateTime expiryDate, int expiryDuration, int numberOfLicenses, ELicenseType licenseType, string machineID, int features)`

Parameters

ExpiryDate - The date when the license will expire. Use NULL if you do not want to specify an expiry date.

ExpiryDuration - The duration of the evaluation period in days. Use -1 if you do not want to specify a duration.

NumberOfLicenses - The number of licenses associated with the key. Use 1 if you do not want to use single activation licensing.

LicenseType - Specify the type of license to generate. See the definition of LicenseType below.

MachineID - A unique identifier for the machine. If you specify a *ComputerName* as the LicenseType, this argument must be the Computer Name. If you specify *User Defined* as the LicenseType, this argument can be anything you want. When validating the license key in your code, you will need to provide the same value to the ValidateLicenseEx function. To create a key that is not computer bound, set this argument to NULL and set the LicenseType to Generic.

Features - A value specifying the features that should be enabled in the created key. Each feature has a unique ID associated to it. To combine features, perform a bitwise OR operation on the required features.

Return

A computer bound license key.

CreateLicenseKeyEx4

Creates a computer bound license key that has an expiry date, a number of licenses and a specific set of features that are enabled.

Prior to calling this function, you must call DefineProduct and set the PrivateKey property. Note that including the PrivateKey in your code is not recommended. Creation of license keys should not typically be done from within the application but rather from a server that the user does not have access to.

C++: `_bstr_t CreateLicenseKeyEx4 (DATE expiryDate, int expiryDuration, int numberOfLicenses, ELicenseType licenseType, BSTR machineID, int[] features)`

C#: `string CreateLicenseKeyEx4 (System.DateTime expiryDate, int expiryDuration, int numberOfLicenses, ELicenseType licenseType, string machineID, SAFEARRAY *Features)`

Parameters

ExpiryDate - The date when the license will expire. Use NULL if you do not want to specify an expiry date.

ExpiryDuration - The duration of the evaluation period in days. Use -1 if you do not want to specify a duration.

NumberOfLicenses - The number of licenses associated with the key. Use 1 if you do not want to use single activation licensing.

LicenseType - Specify the type of license to generate. See the definition of LicenseType below.

MachineID - A unique identifier for the machine. If you specify a *ComputerName* as the LicenseType, this argument must be the Computer Name. If you specify *User Defined* as the LicenseType, this argument can be anything you want. When validating the license key in your code, you will need to provide the same value to the ValidateLicenseEx function. To create a key that is not computer bound, set this argument to NULL and set the LicenseType to Generic.

Features - An array of feature sets. Each feature set is a value specifying the features that should be enabled in the created key. The value of the feature set is the or'ed value of all the features to be enabled in the set. To combine features, perform a bitwise OR operation on the required features.

Return

A computer bound license key.

CreateLicenseKeyEx5

Creates a computer bound license key that has an expiry date, a number of licenses and a specific set of features that are enabled. This API is functionally identical to CreateLicenseKeyEx4. It was created to accomodate programming languages such as VB6 that cannot handle the array data type used in CreateLicenseKeyEx4.

Prior to calling this function, you must call DefineProduct and set the PrivateKey property. Note that including the PrivateKey in your code is not recommended. Creation of license keys should not typically be done from within the application but rather from a server that the user does not have access to.

C++: `_bstr_t CreateLicenseKeyEx5 (DATE expiryDate, int expiryDuration, int numberOfLicenses, ELicenseType licenseType, BSTR machineID, BSTR features)`

C#: `string CreateLicenseKeyEx5 (System.DateTime expiryDate, int expiryDuration, int numberOfLicenses, ELicenseType licenseType, string machineID, string Features)`

Parameters

ExpiryDate - The date when the license will expire. Use NULL if you do not want to specify an expiry date.

ExpiryDuration - The duration of the evaluation period in days. Use -1 if you do not want to specify a duration.

NumberOfLicenses - The number of licenses associated with the key. Use 1 if you do not want to use single activation licensing.

LicenseType - Specify the type of license to generate. See the definition of LicenseType below.

MachineID - A unique identifier for the machine. If you specify a *ComputerName* as the LicenseType, this argument must be the Computer Name. If you specify *User Defined* as the LicenseType, this argument can be anything you want. When validating the license key in your code, you will need to provide the same value to the ValidateLicenseEx function. To create a key that is not computer bound, set this argument to NULL and set the LicenseType to Generic.

Features - A set of features to be enabled using the following syntax:

`<featureSet>:<featureValue>;<featureSet>:<featureValue>`

Example: "0:8;1:2;3:14" - Enables: feature id 8 in feature set 0, feature id 2 in feature set 1 and feature ids 2, 4, 8 ($2 + 4 + 8 = 14$) in feature set 3.

To combine features, perform a bitwise OR operation on the required features.

Return

A computer bound license key.

DaysLeft

Returns the number of days left before the evaluation expires. You must call `ValidateLicense` prior to calling this function.

C++: `int DaysLeft`

C#: `int DaysLeft ()`

Return

Number of days left before the evaluation expires.

DefineProduct

The DefineProduct method initializes basic information required to validate license keys. You must call this function prior to calling any other function.

C++:

```
VARIANT_BOOL DefineProduct ( int ProductID, BSTR ProductName, int MajorVersion, int  
MinorVersion, BSTR EncryptionKey, BSTR PersistenceKey )
```

C#:

```
bool DefineProduct ( int ProductID, string ProductName, int MajorVersion, int MinorVersion, string  
EncryptionKey, string PersistenceKey )
```

Parameters

ProductID

: ID of the product as generated by Quick License Manager

ProductName

: Name of the product

MajorVersion

: Major version of the product (maximum 2 digits)

MinorVersion

: Minor version of the product (maximum 2 digits)

Encryption

Key: string used to encrypt the license key.

PersistenceKey

: GUID associated with the product and automatically generated by Quick License Manager for each product. The evaluation information of the product is stored at runtime in the registry under HKCR\CLSID\<GUID>.

Duration

Returns the duration in days of the evaluation key. You must call `ValidateLicense` prior to calling this function.

C++: `int Duration`

C#: `int Duration ()`

Return

Duration of the evaluation key.

ELicenseStatus

Enum of all possible values of the license key status. Note that the status can consist of a combination of these values:

EKeyPermanent : The license key is valid and it is a permanent license key.

EKeyInvalid : The license key is invalid. It was not decoded successfully.

EKeyDemo : The license key is an evaluation key.

EKeyProductInvalid : The product ID of the license key does not correspond to the expected Product ID.

EKeyVersionInvalid : The Major or Minor version of the license key does not correspond to the expected Major or Minor version.

EKeyExpired : The license key has expired.

EKeyTampered

: The license key was tampered typically indicating that the user is attempting to set the date back to run the software.

EKeyMachineInvalid

: If you are using computer bound license keys, an *EKeyMachineInvalid* status indicates that the license key that was validated does not match the computer to which the license key was bound.

EKeyNeedsActivation

: This flag indicates that the license key is an activation key. If you detect an activation key, you should not enable your application. You should just allow the user to activate his license. Once the license is activated, a computer bound key is issued. Once you detect a valid computer bound key, you can enable your application.

ELicenseType

Enum of all possible types of license keys.

Activation : The license key is a key that requires activation.

Evaluation (obsolete) : The license key is an evaluation key.

ComputerName : The license key is bound to the name of the computer.

Generic (previously PermanentGeneric) : The license key is permanent and not bound to a computer.

UserDefined

: The license key is bound to the computer based on a user defined unique identifier.

EvaluationPerUser

Set this property to true to store evaluation information per user. The default value is true. If set to false, evaluation information is stored at the machine level. Note that you need to make sure the current user has the required privileges to store evaluation information at the machine level under `HKEY_LOCAL_MACHINE\Software\Classes\CLSID\<GUID>`.

Evaluation information consists of the installation date of your software as well as the last time your software ran.

C++: `VARIANT_BOOL EvaluationPerUser`

C#: `bool EvaluationPerUser`

ExpiryDate

Returns the expiry date of the evaluation key. You must call `ValidateLicense` prior to calling this function.

C++: `DATE ExpiryDate`

C#: `System.DateTime ExpiryDate()`

Return

Expiry date of the evaluation key.

Features

Returns an array of all the feature sets associated with the license key. Within a feature set (each element of the array), if several features are associated to a license key, the returned value is a bitwise OR of these features.

This function must be called after a call to `ValidateLicense` or `ValidateLicenseEx`.

C++: `int *Features`

C#:/STRONG> `int [] Features`

GetStatus

Returns the last status. See ELicenseStatus for possible values.

You must always call this function after calling ValidateLicense or ValidateLicenseEx to get the result of the validation.

C++: int GetStatus

C#: int GetStatus ()

Return

Last status

IsEvaluation

Returns whether the current license key is an evaluation key. You must call ValidateLicense prior to calling IsEvaluation.

C++: VARIANT_BOOL IsEvaluation

C#: bool IsEvaluation ()

Return

Boolean indicating if the license key is an evaluation key.

IsFeatureEnabled

Returns whether the specified feature is enabled in this license key. This function is now obsolete and has been superseded by IsFeatureEnabledEx.

You must call ValidateLicense prior to calling IsFeatureEnabled. C++: VARIANT_BOOL

IsFeatureEnabled (int feature)

C#: bool IsFeatureEnabled (int feature)

Parameters

feature - id of feature to be checked.

Return

Boolean indicating if the featured is enabled.

IsFeatureEnabledEx

Returns whether the specified feature is enabled in this license key. You must call `ValidateLicense` prior to calling `IsFeatureEnabled`.

C++: `VARIANT_BOOL IsFeatureEnabled (int featureSet, int feature)`

C#: `bool IsFeatureEnabledEx (int featureSet, int feature)`

Parameters

featureSet - id of the feature set. QLM supports four feature sets (0 to 3).

feature - id of feature to be checked.

Return

Boolean indicating if the featured is enabled.

IsValid

Returns whether the current license key is a valid key. A valid license key is a key that was decoded properly and is either permanent or evaluation. You must call `ValidateLicense` prior to calling `IsValid`.

C++: `VARIANT_BOOL IsValid`

C#: `bool IsValid()`

Return

Boolean indicating if the license key is a valid key.

LicenseType

Returns the license type of the key. See ELicenseType for possible values.

C++: ELicenseType LicenseType

C#: ELicenseType LicenseType ()

Return

License type

NumberOfLicenses

Returns the number of multiple activations enabled for the license key.

C++: int NumberOfLicenses

C#: int NumberOfLicenses;

Return

Number Of LNumber Of Licenses

MajorVersion

Returns the major version associated to the license key. You must call `ValidateLicense` prior to calling this function.

C++: `int MajorVersion`

C#: `int MajorVersion`

Return

Major version of the product.

MinorVersion

Returns the minor version associated to the license key. You must call ValidateLicense prior to calling this function.

C++: int MinorVersion

C#: int MinorVersion

Return

Minor version of the product.

PrivateKey

QLM version 5 implements asymmetric encryption to encrypt the license key. Asymmetric encryption is safer because one key encrypts the license, the private key, and another key, the public key, decrypts that information. Therefore, you only need to include the public key in your source code.

This function sets the private key associated with your product. The private key should be set before you create a license, typically right after the call to DefineProduct. If you are creating a license key with a QLM engine version prior to version 5, you do not need to set the private key. It is highly recommended that you do not set the private key in your code.

The private key of your product can be found on the DefineProduct screen under the Keys tab in the QLM Console.

C++: `_bstr_t privateKey`

C#: `string PrivateKey`

PublicKey

QLM version 5 implements asymmetric encryption to encrypt the license key. Asymmetric encryption is safer because one key encrypts the license, the private key, and another key, the public key, decrypts that information. Therefore, you only need to include the public key in your source code.

This function sets the public key associated with your product. The public key should be set before you validate a license, typically right after the call to `DefineProduct`. If you are validating a license key with a QLM engine version prior to version 5, you do not need to set the public key.

The public key of your product can be found on the `DefineProduct` screen under the `Keys` tab in the QLM Console.

C++: `_bstr_t publicKey`

C#: `string PublicKey`

ProductID

Returns the product ID associated to the license key. You must call ValidateLicense prior to calling this function.

C++: ibt ProductID

C#: int ProductID ()

Return

Product ID associated to the license key.

ValidateLicense

Validates a license key. You must call DefineProduct prior to calling this function. After calling this function, call GetStatus to get the status of the call.

C++: `_bstr_t ValidateLicense (BSTR LicenseKey);`

C#: `string ValidateLicense (string LicenseKey)`

Parameters

LicenseKey

: License Key to validate

Return

Error message if ValidateLicense fails to validate or if the license is an evaluation license.

ValidateLicenseEx

Validates a computer bound license key. You can call this function for any type of license key. If the license key is not computer bound, set the ComputerID to an empty string. You must call DefineProduct prior to calling this function.

After calling this function, call GetStatus to get the status of the call.

C++: `_bstr_t ValidateLicenseEx (BSTR LicenseKey, BSTR ComputerID);`

C#: `string ValidateLicenseEx (string LicenseKey, string ComputerID)`

Parameters

LicenseKey

: License Key to validate

ComputerID

: A string identifying the computer.

Return

Error message if ValidateLicenseEx fails to validate or if the license is an evaluation license.

ValidateLicenseEx2

Validates a computer bound license key. You can call this function for any type of license key. If the license key is not computer bound, set the ComputerID to an empty string. You must call DefineProduct prior to calling this function.

After calling this function, call GetStatus to get the status of the call.

C#: string ValidateLicenseEx2 (string licenseKey, string computerID, bool skipWrites, bool skipExpiryValidation)

Parameters

LicenseKey

: License Key to validate

computerID

: A string identifying the computer.

skipWrites

: Do not write any data on the end user system. Should be set to false in most cases.

skipExpiryValidation

:Do not check for expiry of the license. Should be set to false in most cases.

Return

Error message if ValidateLicenseEx2 fails to validate or if the license is an evaluation license.

ValidateLicenseEx3

Validates a computer bound license key. You can call this function for any type of license key. If the license key is not computer bound, set the LicenseBinding to None. You must call DefineProduct prior to calling this function.

After calling this function, call GetStatus to get the status of the call.

C#: string ValidateLicenseEx3 (string LicenseKey, ELicenseBinding licenseBinding, bool skipWrites, bool skipExpiryValidation)

Parameters

LicenseKey

: License Key to validate

licenseBinding

: License Binding Type

skipWrites

: Do not write any data on the end user system. Should be set to false in most cases.

skipExpiryValidation

: Do not check for expiry of the license. Should be set to false in most cases.

Return

Error message if ValidateLicenseEx3 fails to validate or if the license is an evaluation license.

ValidateFile

Validates that the Quick License Manager DLL is authentic and was not tampered with. In order to prevent hackers from replacing the IsLicense50.dll with their own version, you can validate the authenticity of the DLL by calling the ValidateFile function. The ValidateFile function returns a fingerprint (long number) that is the result of a checksum of the DLL contents combined with your own key. Use the QlmFingerPrint.exe to generate this unique fingerprint and validate in your code that the runtime fingerprint matches the generated one.

If you are using QLM Professional, you do not need to call this function. Instead, set the validateIntegrity argument to true when constructing the QlmLicense object.

C++: long ValidateFile (BSTR LicenseDLL, BSTR Key);

C#: ulong ValidateFile(string LicenseDLL, BSTR Key)

Parameters

LicenseDLL

: Full path to the License DLL. If this argument is NULL, the currently loaded License DLL is used.

Key

: A unique key of your choice that is used to uniquely encrypt the fingerprint.

Return

FingerPrint- A long number that uniquely identifies your license DLL.

Version

Returns the version of the QLM engine used to create the key. You must call `ValidateLicense` prior to calling this function.

C++: `_bstr_t` Version

C#: `string` Version

Return

Version of QLM Engine used to create the license key.

Quick License Manager Professional Overview

Quick License Manager Professional provides the tools required to implement online software activation. QLM Professional is composed of 3 components:

- A License Server / database that exposes an interface for issuing and managing license keys
- A Windows client application (QLM Management Console) that communicates with the License Server and allows you to manage license keys.
- A set of APIs and controls that you can use and integrate in your application.

Software Activation is the process of generating a computer bound license key over the internet. In order to implement software activation with Quick License Manager, you need to work with 2 types of license keys.

The first key is called "Activation Key" (ELicenseType.Activation). This is the key that you send to your customer when they purchase your software. The activation key can be sent to your customer directly through our integration with leading [eCommerce providers](#)

. It can also be sent from the QLM Management Console or by any other means of your choice. This license key does not enable your software. It simply allows the user to activate his license. If a user purchases several copies of your software, you can send them one activation key for all purchased licenses or one activation key per license.

From your application, the user enters the activation key. Your application then calls the QLM API and sends the activation key along with a computer unique identifier to the QLM License Server. The QLM License Server generates the second type of license key called "Computer Bound Key" (ELicenseType.UserDefined or ELicenseType.ComputerName) and sends this key back to your application. The computer bound key enables your application to run.

Note that in your application, you need to have a dialog for the user to enter a license key and activate it. QLM includes [.NET Controls](#) that you simply drop in your application to handle the license activation part. For non .NET applications, QLM provides the [QLM License Wizard](#) which is a standalone executable that you can launch from your application.

License Server

The QLM License Server provides an interface to the database that stores all license keys and related data. The default database that ships with QLM is a MS-Access database. The supported databases are MS-SQL Server 2000 and higher and MS-Access. See the [Configure the Database](#) section for more details on installing the database.

The system requirements for the QLM License Server are:

- Windows 2008 server or higher (x86 or x64).
- .NET Framework 4.0
- SQL Server Database or MS-Access
- Full Trust for .NET assemblies.

The License Server can be installed in 2 ways: (a) by running the provided setup program

QlmLicenseServerSetup.exe

or (b) by executing the installation steps manually. If you are hosting your own web site or if you are distributing the QLM License Server as part of your program (requires special distribution rights and a QLM Enterprise license), then option (a) is recommended. If your site is hosted at an ISP, then you may need to use option (b).

Automated installation of the License Server

To install the QLM License Server, locate the **QlmLicenseServerSetup.exe** setup program in the QLM installation folder. Typically this file is located under:

%Public%\Public Documents\Quick License Manager\DeployToServer Execute the setup and follow the onscreen instructions.

Manual installation of the License Server

To manually install the QLM License Server, locate the **QlmLicenseServer** folder in the QLM installation folder. Typically this folder is located under:

%Public%\Public Documents\Quick License Manager\DeployToServer\QlmLicenseServer

- At your ISP, create a new virtual directory called **qlm** and enable ASP.NET 4.0 for this virtual directory.
- Create an Application Pool and associate the virtual directory above to the Application Pool.
- Ensure the Application Pool is configured for .NET 4.0
- Upload all the files in the %Public%\Public Documents\Quick License Manager\DeployToServer\QlmLicenseServer folder and subfolders to the virtual directory (preserve the directory structure).

Configure the License Server Customize the following settings in the web.config files based on your needs:

- Database connection string. Refer to the [Configure the Database](#) section in the Help for instructions on installing the database.
- Default QLM Engine Version (defaultQlmVersion).
- SQL Syntax(sqlsyntax).
- Communication Encryption Key (communicationEncryptionKey). The communication encryption key is used to encrypt data transferred between QLM and the QLM License Server. This key is like a password that protects your data.
- Admin Encryption Key (adminEncryptionKey). The admin encryption key is used to encrypt data transferred between QLM and the QLM License Server. This key is like a password that protects your data.

Security Note:

- You need to give the anonymous user (IUSR_XXX, IWAM_XXX) execute privileges to the bin folder.

Recommendations

- Change the default Communication and Admin Encryption Keys. If you do not, any other QLM customer may be able to view your data.

Configuring the Database

QLM Professional stores all issued license keys as well as customer related information in a database on the web server. The default database that ships with QLM is a MS-Access database. However, you can use any database that implements an OleDbProvider.

Database Installation

If you installed the QLM License Server using the provided setup program, the database is created automatically during the setup.

However, if you manually installed the QLM License Server, you will need to create the database manually as well as follows:

- If you want to use the MS-Access Database, copy the qlm.accdb file from %Public%\Public Documents\Quick License Manager\DeployToServer\QlmLicenseServer\Db to the location recommended by your ISP and update the web.config file accordingly. Note that the web.config must contain the appropriate local path to the qlm.accdb file.
- If you want to use an SQL Server database, use the tools provided by your ISP to create a database called Qlm (or any other name of your choice) or execute the sql200x.createdb.sql.
- Execute the provided database creation script sql200x.createtables.sql.
- Execute the provided database creation script sql200x.createusers.sql.
- Update the web.config file to point to the SQL database (see comments in web.config and section below).

Configure QLM to use a SQL Server Database

To use a database engine other than MS-Access, you need to update the connectionString and the sql Syntax settings in the web.config file that is included with the License Server as follows:

For all SQL Server editions, locate the sqlSyntax setting in the web.config and set it as follows:

```
<setting name="sqlSyntax" serializeAs="String">  
  <value>sql</value>  
</settings>
```

For the connectionStrings, locate in the web.config a commented connectionString section that corresponds to the type of database you are using. Uncomment the section and update the connectionString settings accordingly.

Recommendations

- Backup your database on a daily basis.

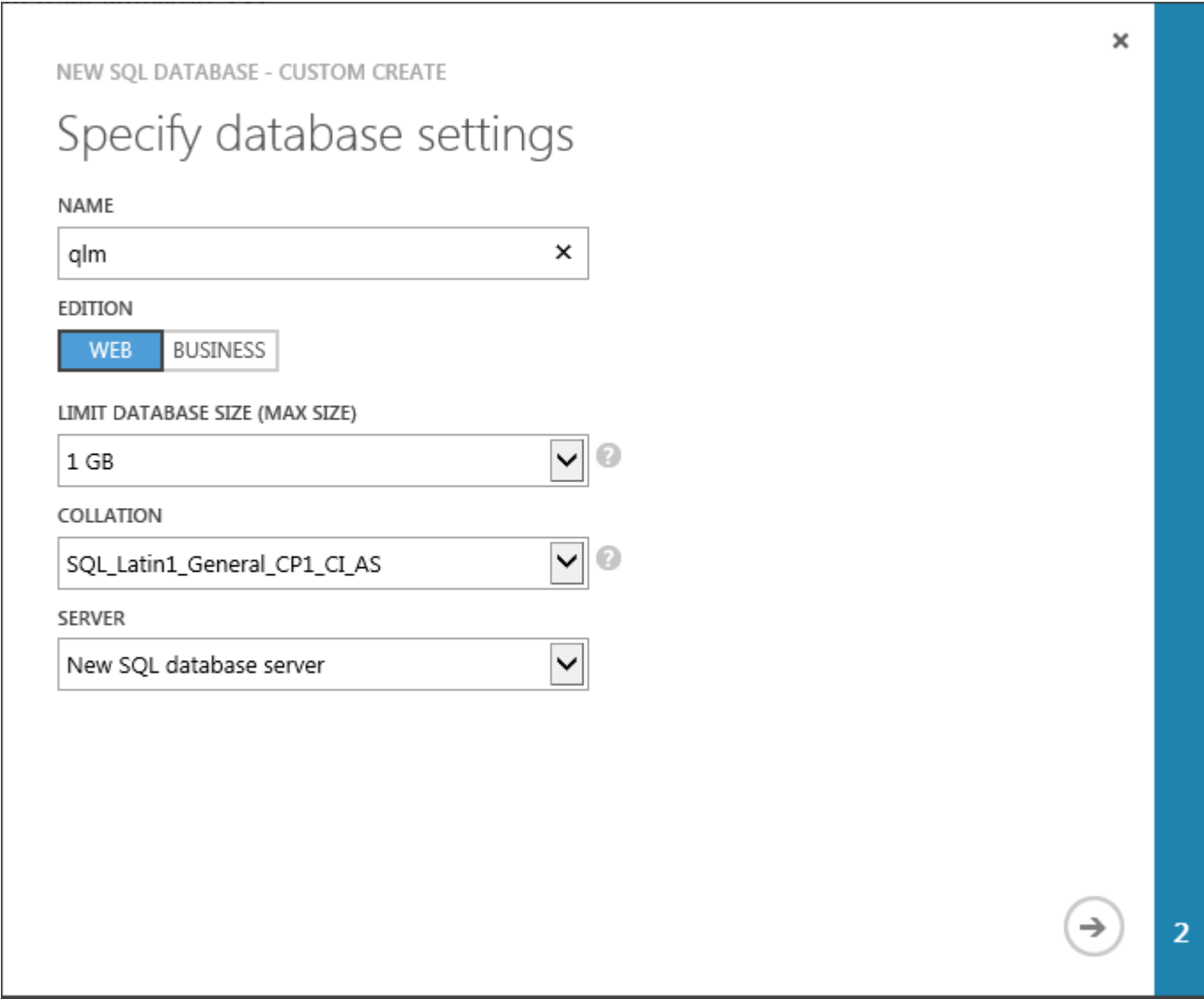
QLM and Windows Azure Integration

You can host the QLM License Server on a Windows Azure portal. The procedure outlined in this section will show you how to build a QLM Azure deployment package, how to create the database on the Azure Portal and finally how to connect QLM to the Azure hosted QLM License Server.

1. Database Creation

To create the QLM database on the Azure portal:

Go to the Windows Azure Portal . Select SQL Database s. Click on the New button . Click on Quick Create. Set the



database name to: **qlm**
Pick the server of your choice or create a new SQL database server.
Click on **Create SQL database**.

2. Server settings

If you selected to create a new SQL database server on the previous step, you need to configure a user

account on the SQL Server.

Enter qlm as the login name of the server
Enter a password
Select a region
When done, confirm that you see the database in the list of available databases.

NEW SQL DATABASE - CUSTOM CREATE

SQL database server settings

LOGIN NAME

qlm

LOGIN PASSWORD

.....

CONFIRM PASSWORD

.....

REGION

Southeast Asia

☒ ALLOW WINDOWS AZURE SERVICES TO ACCESS THE SERVER.

1

sql databases


DATABASES SERVERS


NAME	STATUS	LOCATION	SUBSCRIPTION	SERVER	EDITION
qlm	→ ✓ Online	Southeast Asia	Free Trial	o5ad7dxk8e	Web


3. Configure the database

Now that the database is created, we need to create the tables and stored procedures. Follow the steps below:

Click on the QLM database in the list. In the Connect to your database panel, click on "Run Transact-SQL queries against your SQL database".




 DASHBOARD MONITOR SCALE CONFIGURE



You created a new SQL database


Here are a few options to get you started

☐ Skip Quick Start the next time I visit




Get Microsoft database design tools ?

[Install Microsoft SQL Server Data Tools](#)



Design your SQL database ?

[Download a starter project for your SQL database](#) [Set up Windows Azure firewall rules for](#)



Connect to your database ?

[Design your SQL database](#) [Run Transact-SQL queries against your SQL database](#)

strings for ADO .Net, ODBC, PHP, and JDBC

Server: o5ad7dxk8e.database.windows.net,1433

ab
ase
.

W
he
n
pr
om
pte
d
to
log
in,
ent
er
yo
ur
S
Q
L
Se
rve
r
cre
de
nti
als.



SQL DATABASE

SERVER

o5ad7dxk8e.database.windows.net

DATABASE

qlm

USERNAME

qlm

PASSWORD

••••••••

Connecting...

Log on →

Cancel

Click on New Query and paste the content of the file: %Public%\Documents\QuickLicense Manager\DeployToAzure\Sql.m.createables.sql

```
New Query Open Save As Run Actual Plan Estimate... Stop

/***** Object: Table [LicenseKeys]    Script Date: 08/11/2007 22:56:43 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
IF NOT EXISTS (SELECT * FROM sys.objects WHERE object_id = OBJECT_ID(N'[LicenseKeys]') AND type in (
BEGIN
CREATE TABLE [LicenseKeys](
    [ID] [int] IDENTITY(1,1) NOT NULL,
    [ActivationKey] [nvarchar](100) NULL,
    [ComputerKey] [nvarchar](100) NULL,
    [ComputerID] [nvarchar](50) NULL,
    [UserID] [int] NULL DEFAULT ((0)),
    [ProductID] [int] NULL DEFAULT ((3)),
    [MajorVersion] [nvarchar](50) NULL DEFAULT ('2'),
    [MinorVersion] [nvarchar](50) NULL DEFAULT ('4'),
    [OrderDate] [datetime] NULL,
    [ActivationDate] [datetime] NULL,
    [LastAccessedDate] [datetime] NULL,
    [ActivationCount] [int] NULL DEFAULT ((0)),
    [OrderID] [nvarchar](50) NULL,
    [Comment] [nvarchar](255) NULL,
    [MaintenanceRenewalDate] [datetime] NULL,
    [MaintenancePlanNotification] [bit] NULL,
    [SubscriptionExpiryDate] [datetime] NULL,
    [GenericLicense] [bit] NULL DEFAULT ((0)),
    [ReleaseCount] [int] NULL DEFAULT ((0)),
    [ReleaseDate] [datetime] NULL,
    [NumLicenses] [int] NULL DEFAULT ((0)),
    [AvailableLicenses] [int] NULL DEFAULT ((0)),
    [ComputerName] [nvarchar](100) NULL,
    [ClientVersion] [nvarchar](50) NULL,
    [Disabled] [bit] NULL DEFAULT ((0)),
    [UserData1] [nvarchar](3000) NULL,
    [AffiliateID] [nvarchar](32) NULL,
    [ReceiptID] [nvarchar](50) NULL,
    [OrderStatus] [bigint] NULL,
    [FloatingSeats] [int] NULL DEFAULT ((0)),
    [FloatingLicenseLocation] [nvarchar](255) NULL
)

/***** Object: Index [ActivationKey]    Script Date: 08/11/2007 23:02:36 *****/
CREATE CLUSTERED INDEX [ActivationKey] ON [LicenseKeys]
(
    [ActivationKey] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, SORT_IN_TEMPDB = OFF, IGNORE_DUP_KEY = OFF,

/***** Object: Index [User]    Script Date: 08/11/2007 23:02:47 *****/
CREATE NONCLUSTERED INDEX [User] ON [LicenseKeys]
(
    [UserID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, SORT_IN_TEMPDB = OFF, IGNORE_DUP_KEY = OFF,
```

Click on Run to execute the query Repeat the steps for the following files, in the exact order der listed here: Install Commons ql, Install Roles.

sql
,
Ins
tall
Pe
rso
nali
zati
on.
sql
,
Ins
tall
Pr
ofil
e.s
ql,
Ins
tall
M
em
ber
shi
p.s
ql

4. Creating the QLM package

To deploy the QLM License Server to Windows Azure, you must create an Azure package by following the steps outlined below:

La
un
ch
the
Q
L
M
M
an
ag
em
ent
Co
ns
ole
an
d
go
to
the

Database connection

Set the fields below if you are manually uploading the QLM web service to your server. If you are using the qlmwebsvcsetup.exe to deploy the QLM web service, you do not need to set these fields.

Server Hostname or IP:

o5ad7dxk8e.database.windows.net


Database Name:
(required for SQL Server only)

qlm

User Name:

qlm

User Password:

 Update config files

M
an
ag
e
Ke
ys
tab
Cli
ck
on
the
Sit
es
but
ton
in
the
too
lba
r,
the
n
clic
k
on
Ad
d
Ent
er
a
na
me
for
yo
ur
Az
ure
site
,
say
Ql
m
Az
ure
,
an
d
clic
k
O

K.
In
the
Pri
ma
ry
Sit
e
fiel
d,
ent
er
(yo
u
ca
n
rep
lac
e
'm
yql
m'
wit
h
an
y
val
ue
of
yo
ur
ch
oic
e):
htt
p://
my
ql
m.
clo
ud
ap
p.n
et/
ql
mli
ce
nse
ser
ver

/Ql
mS
erv
ice
.as
mx
If
yo
u
rep
lac
ed
'm
yql
m'
wit
h
an
y
oth
er
val
ue,
not
e
tha
t
yo
u
will
ne
ed
to
use
the
sa
me
val
ue
lat
er
on
in
this
pr
oc
ed
ure
.
Set

the
Da
tab
ase
En
gin
e
to:
S
Q
L
Se
rve
r
Cli
ck
on
the
En
cry
pti
on
Ke
ys
tab
,
the
n
clic
k
Ne
w
for
bot
h
fiel
ds
to
ge
ner
ate
a
Co
m
mu
nic
ati
on
En
cry

pti
on
Ke
y
an
d
an
Ad
mi
nE
ncr
ypt
ion
Ke
y
Cli
ck
on
the
Da
tab
ase
Co
nn
ect
ion
tab
an
d
ent
er
all
the
fiel
ds
on
this
tab
Se
rve
r
Ho
stn
am
e
or
IP:
to
get
this

value, select your database in Azure, and click on the Dashboard link. The SERVER NAME field is displayed in the right hand panel

an
d
typ
ical
ly
loo
ks
lik
e:
o5
ad
7d
xk
8e.
dat
ab
ase
.wi
nd
ow
s.n
et
Da
tab
ase
Na
me
:
ql
m
or
an
y
oth
er
na
me
yo
u
ha
ve
ent
ere
d
ear
lier
in
the
pr
oc

ess
Us
er
Na
me
:
na
me
of
the
use
r
yo
u
sp
eci
fie
d
ear
lier
in
the
pr
oc
ess
Us
er
Pa
ss
wo
rd:
pa
ss
wo
rd
of
the
use
r
yo
u
sp
eci
fie
d
ear
lier
in
the
pr

oc
ess
Cli
ck
on
Up
dat
e
co
nfi
g
file
s
Do
not
clie
k
O
K
--
lea
ve
the
Sit
es
edi
tor
op
en.

Th
e
ne
xt
ste
ps
inv
olv
e
bui
ldi
ng
the
pa
ck
ag
e
usi
ng
Vis

ual
Stu
dio
20
12
Op
en
an
d
Bui
ld
the
Ql
m
Az
ure
sol
uti
on
loc
ate
d
in
the
%
Pu
bli
c%
\\D
oc
um
ent
s\
Qu
ick
Lic
ens
e
M
an
ag
er\
De
plo
yT
oA
zur
e\
fol
der

usi
ng
Vis
ual
Stu
dio
20
12
Rig
ht
mo
use
clic
k
on
the
Ql
m
Az
ure
pr
oje
ct
an
d
sel
ect
"P
ac
ka
ge"
Set
the
Se
rvi
ce
Co
nfi
gur
ati
on
to
Cl
ou
d
an
d
Bui
ld
Co

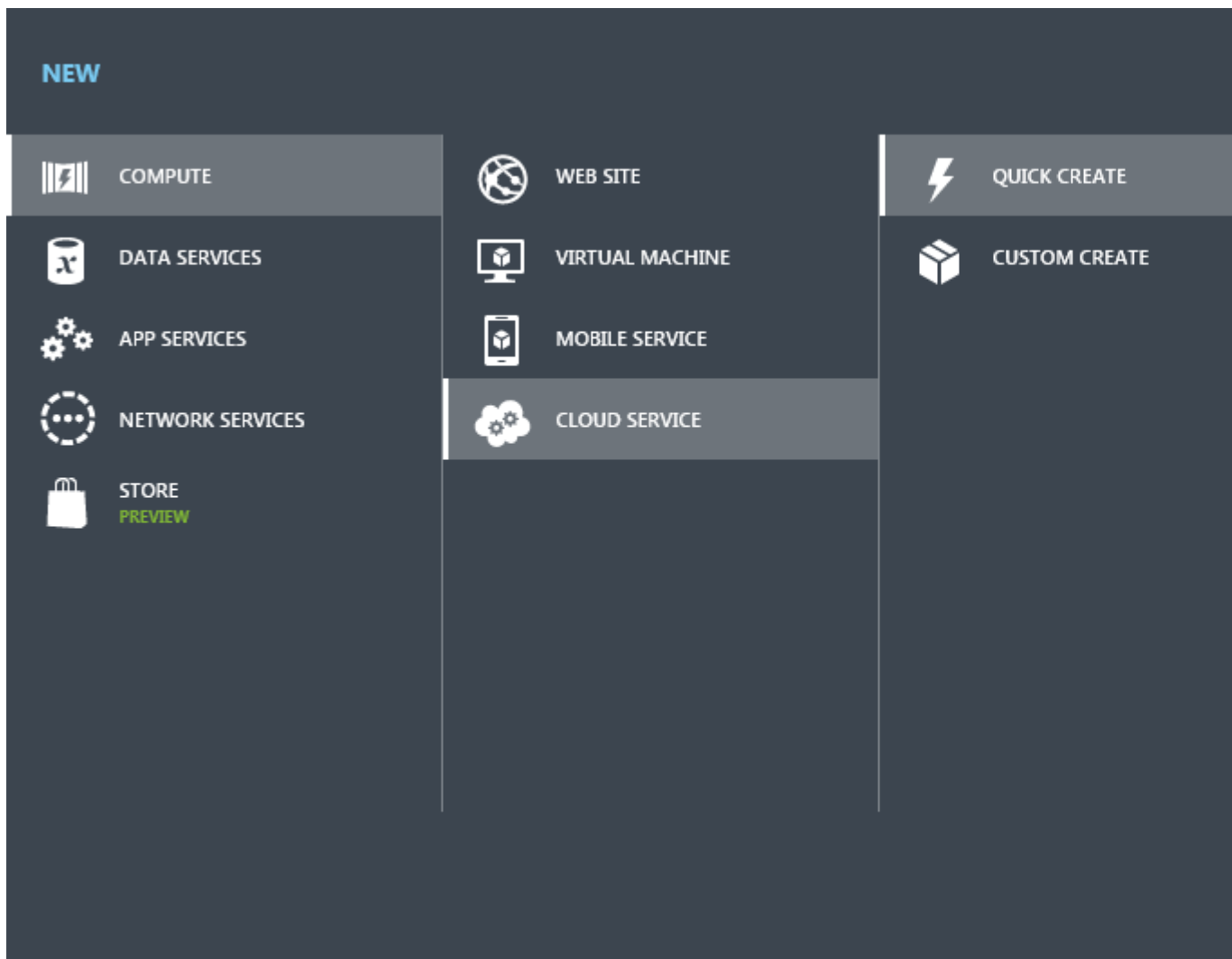
nfi
gur
ati
on
to
Rel
eas
e
the
n
clie
k
on
Pa
ck
ag
e
Th
e
foll
ow
ing
2
file
s
will
be
cre
ate
d
in
the
Q1
m
Az
ure
\Q1
m
Az
ure
\bi
n\
Rel
eas
e\A
pp.
pu
blis
h
fol

der
:
Ql
m
Az
ure
.cs
pk
g
an
d
Se
rvi
ce
Co
nfi
gur
ati
on.
Cl
ou
d.c
scf
g
file
s

1. Cloud Service

Finally, we need to create the Azure Cloud Service and deploy the QLM package to the Azure portal:

Go to the Windows Azure Portal . Select Cloud Services. Click on the New button . Click on Quick Create. Set the URL to any value of yo



ur
ch
oic
e
tha
t
ma
tch
es
the
val
ue
yo
u
set
ear
lier
on
Sit
es
pa
ge:
my
ql
m
Set
the
reg
ion
to
an
y
val
ue
of
yo
ur
ch
oic
e.
Cli
ck
on
Cr
ea
te
Cl
ou
d
Se

rv
ce.

On
ce
cre
ate
d,
clic
k
on
the
ser
vic
e
an
d
sel
ect
Up
loa
d a
ne
w
pr
od
uct
ion
de
plo
ym
en
t.

In
the
De
plo
ym
ent
lab
el
fiel
d,
ent
er:
Q
L
M

In
the

Upload a package.

This will create a new **production** deployment.

DEPLOYMENT LABEL

PACKAGE



FROM LOCAL



FROM STORAGE

CONFIGURATION



FROM LOCAL



FROM STORAGE

☒ Deploy even if one or more roles contain a single instance. [?](#)

☒ Start deployment

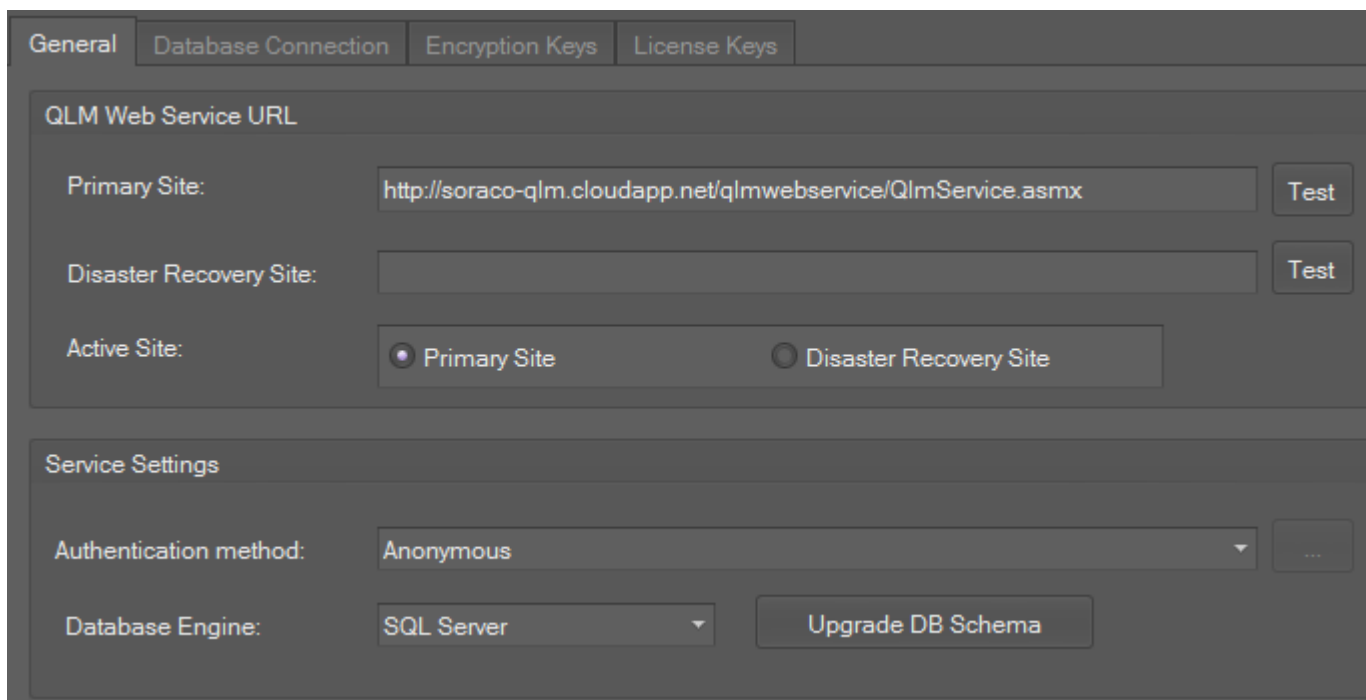


Pa
ck
ag
e
fiel
d,
clie
k
**Fr
om
Lo
cal**
an
d
sel
ect
the
Ql
m
Az
ure
.cs
pk
g
file
cre
ate
d
ear
lier
In
the
Co
nfi
gur
ati
on
fiel
d,
clie
k
**Fr
om
Lo
cal**
an
d
sel
ect
the

Se
rvi
ce.
Co
nfi
gur
ati
on.
Cl
ou
d.c
scf
g
file
cre
ate
d
ear
lier
Ch
ec
k
"D
epl
oy
ev
en
if
on
e
or
mo
re
rol
es
co
nta
in
a
sin
gle
inst
an
ce"
.
Ch
ec
k
"St
art

deployment". Click on the checkmark at the bottom of the page

Once the package is created, click on Dashboard link to view details about the



The screenshot shows a configuration window with four tabs: General, Database Connection, Encryption Keys, and License Keys. The General tab is active. It contains two main sections: 'QLM Web Service URL' and 'Service Settings'.

QLM Web Service URL

- Primary Site:** A text field containing 'http://soraco-qlm.cloudapp.net/qlmwebservice/QLmService.asmx' with a 'Test' button to its right.
- Disaster Recovery Site:** An empty text field with a 'Test' button to its right.
- Active Site:** A section with two radio buttons: 'Primary Site' (which is selected) and 'Disaster Recovery Site'.

Service Settings

- Authentication method:** A dropdown menu set to 'Anonymous' with a three-dot menu button to its right.
- Database Engine:** A dropdown menu set to 'SQL Server' with an 'Upgrade DB Schema' button to its right.

pa
ck
ag
e.
Ta
ke
not
e
of
the
the
Sit
e
Ur
I
No
w
go
ba
ck
to
the
Q
L
M
M
an
ag
em
ent
Co
ns
ole
,
M
an
ag
e
Ke
ys
/
Sit
es
wh
ich
yo
u
left
op
en
ear

lier
.
On
the
Ge
ner
al
tab
,
set
the
Pri
ma
ry
Sit
e
to:
<si
te
Url
>/
ql
mli
ce
nse
ser
ver
/ql
ms
erv
ice
.as
mx
Cli
ck
the
**Te
st**
but
ton
ne
xt
to
the
pri
ma
ry
site
U
RL

an
d
co
nfir
m
tha
t
all
tes
ts
ha
ve
pa
sse
d.

This completes the configuration of the Azure site.

Disaster Recovery

When configuring a License Server site in the QLM Console, you can configure a disaster recovery (DR) site where your data will be replicated. The DR site can take over processing requests should the primary site fail. To configure a DR site, you must:

- Install the QLM License Server on the DR server
- Install the QLM database on the DR server
- Edit the License Server profile in the QLM Console and add a URL to the QLM DR Site.
- Setup a backup job in QLM to backup the database on a daily basis (or as often as required). To setup a backup job, click on the Backup tab then click on "Create Backup".
- Configure the backup job to automatically restore data to the DR site when the backup is completed. This is done by checking the AutoRestore checkbox when configuring the backup.

Once the above steps are completed, all keys stored on the primary site will be restored to the DR site after the backup completes.

QLM License Server redirection for your application

QLM also supports redirection of the QLM License Server to a disaster recovery site in your application. This will allow your customers to successfully contact the DR site when the primary site is down.

Your application's failover

Failover to a disaster recovery site is controlled via an XML file that you place on your server. A sample XML file is located in the QLM installation folder and is called `qlmredirect.xml`. The xml file contains the URL to the active QLM License Server. If your primary QLM site is down and you want to fail over to the DR site, you need to change the URL in the `qlmredirect.xml` to point to your DR site. Note that the `qlmredirect.xml` file should never be located at your primary site.

Failover is purposely non-automated. If failover was automatic, small network glitches could trigger a failover and cause some data to be stored on the primary site and other data on the DR site. QLM does not support merging data that has been updated on both ends.

Code changes to enable redirection in your application

In your code, when initializing the `QlmLicense` object, you must set 2 properties:

- `EnableDRSite`
- `RedirectorUrl`

To enable redirection, set `EnableDRSite` to true. To specify the URL to the `qlmredirect.xml` file, set the `RedirectorUrl` to the URL that points to this file.

For example:

```
QlmLicense license = new QlmLicense ();  
license.DefineProduct (...);  
license.EnableDRSite = true;  
license.RedirectorUrl = "http://soraco.co/qlmredirect.xml";
```

Note that after failing to a DR site, new data will be written to the DR site. When your primary site is online again, you may want to replicate all the data on the DR site to the Primary Site. This can be done from the Backup / Restore tab. Note that QLM does not merge data when restoring. All data is overwritten from the backup. Exercise extreme caution when performing restores. It is highly recommended that you perform your own database backup before restoring data to the QLM database.

Multiple Activations Keys

When customers purchase several copies of your product, there are two ways you can send them their license keys.

1. You can send them one license key for each purchased copy. For example, if a customer purchases 5 copies of your software, you will send them 5 license keys. This approach is inconvenient both for the seller who has to manage multiple keys per customer and the buyer who has to associate a key per computer.
2. You can send them one license key that can be activated on 5 computers. This approach is simple both for the vendor and the seller. This method is referred to in QLM as **Multiple Activations Key**.

How To create a license key using Multiple Activations

From the QLM Console, click on Manage Keys / License Keys / Create. In the Create License dialog, select the Multiple Activations key check box and specify the number of activations.

If you are using an http request such as GetActivationKey to create activation keys, add the `is_usemultipleactivationskey=true` argument to the URL. When integrated with an eCommerce provider, the number of activations is typically determined based on the number of copies purchased.

Alternatively, you can add the `is_quantity` argument in the URL request.

When a license key using Multiple Activations is activated, QLM maintains the activation information in a separate table called ActivationLog.

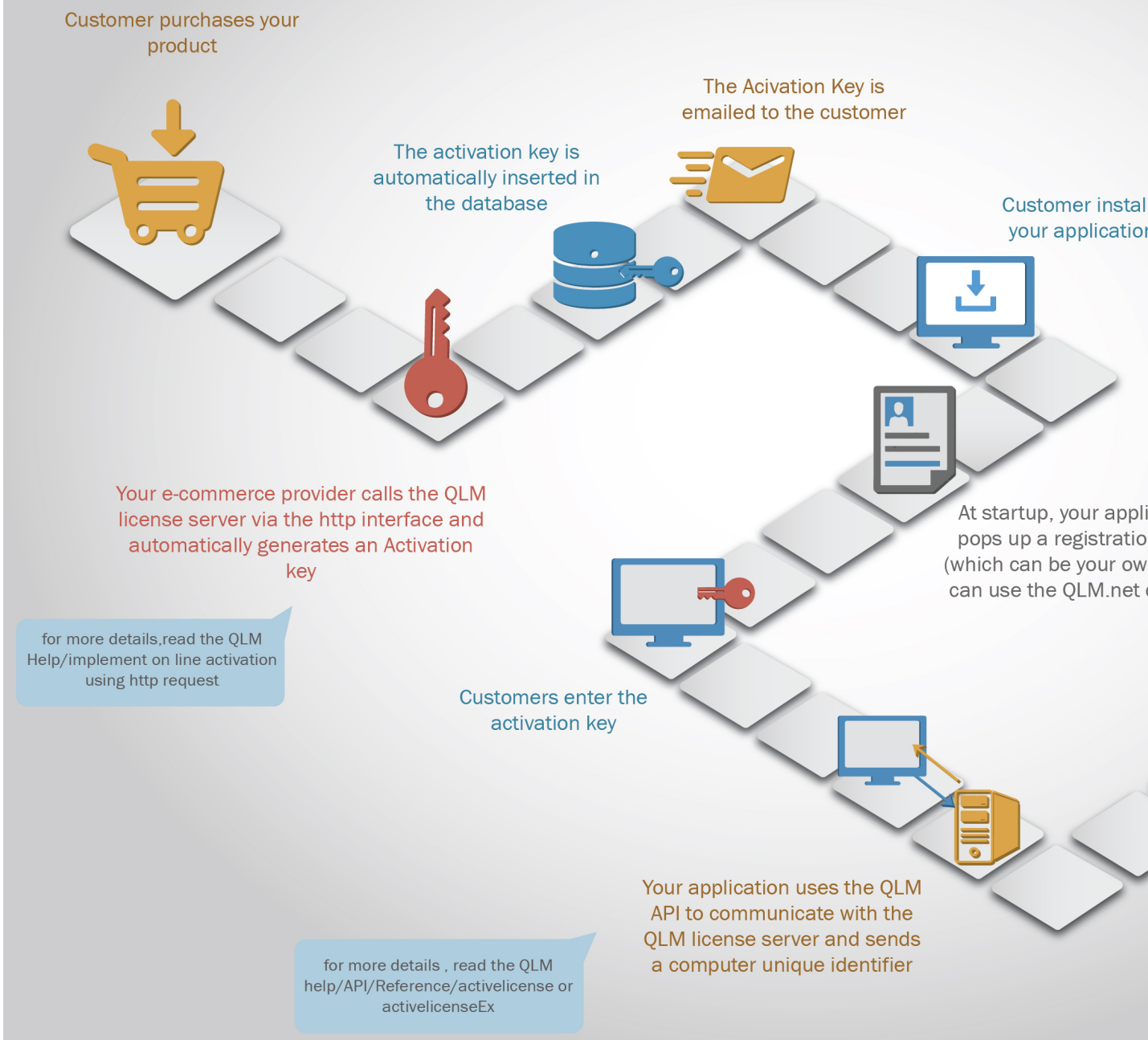
To view the activated licenses, start QLM, click on Manage Keys, run a search to display a set of records. Licenses that are of type Multiple Activations display a + sign in the License Type field. Click on the + sign to expand the row and view the activated licenses.

You can perform one of three operations on the multiple activations licenses:

Release	Releases an activated license.
Edit	Edits an activated license.
Delete	Deletes an activated license.

Fully Automated Workflow with eCommerce Provider Integration

QUICK LICENSE MANAGER - ONLINE ACTIVATION



Partially Automated Workflow with manual eCommerce Provider Integration

QUICK LICENSE MANAGER - ONLINE AC

[Use this method if your eCommerce provider can not call a web serv

Using the QLM Management console, you generate a batch of Activation Keys and upload these keys to your eCommerce Provider

Customer installs your application

At startup, your application pops up a registration window (which can be customized) and the user can use the QLM

Your customer purchases your product. Your eCommerce provider issues one of the available Activation Keys and sends it by email to the customer

Customer enters the Activation Key as well as his contact information

for more details , read the QLM Web Help/API Reference/AddUser and ActiveLicenseForUser

Your application uses the QLM API to communicate with the QLM license server and sends a computer unique identifier along with the customer contact information

Integrate QLM with your application

QLM provides several approaches to protect your application depending on the platform, programming language as well as the desired licensing model. The sections below describe the most common approaches to protect your application. If your requirements are not covered in this section, contact our technical support for professional advice on how to protect your application.

In general, there are 2 steps required to protect your application:

- The first time the user launches your application, you need to present the user with a license registration form so that they can enter a license key and activate it.
- On subsequent launches of your application, you read the previously activated license key and validate it before opening up your application.

Step by Step Procedure to integrate QLM in your application

- [Define a Product](#) in the QLM Management Console.
- [Install](#) the QLM License Server on your server (or Soraco's server if you are using our hosting service) and upload your products to the QLM License Server(skip this step if you are using QLM Express).
- [Configure](#) the QLM Management Console to connect to the QLM License Server (skip this step if you are using QLM Express).
- For QLM Pro and Enterprise users, create a license key from the [Manage Keys tab](#).
- For QLM Express users, create a license key from the [Generate Keys](#) tab.
- Modify your application as described in the [Integrate QLM with your application section](#)
- Make sure to include the following DLLs in your application: QlmLicenseLib.dll, x86\IsLicense50.dll and x64\IsLicense50.dll. For more details, review the [redistributables section](#).

We also provide several samples in a multitude of programming languages. To locate the sample that matches your application, launch the [Get Started Wizard](#) and follow the instructions to locate the proper sample.

Integrate QLM with your .NET WinForms application

For Windows Forms .NET applications, QLM provides 3 .NET Controls that you can easily drop in your application. These controls are forms that allow the end-user to enter a license key and activate it.

The 3 .NET controls are:

- QLM Express:
QlmExpressLicenseValidationControl
(QlmControls.dll)
- QLM Pro/Enterprise:
QlmWebBasicActivationControl
(QlmControls.dll)
- QLM Pro/Enterprise: QlmLicenseWizardCtrl
(QlmControlLicenseWizard.dll).

Both QLM Pro/enterprise controls offer very similar functionality. The main difference between these 2 controls is that the QlmLicenseWizardCtrl uses a wizard based graphical user interface. In addition, the QlmLicenseWizardCtrl can read its properties from 2 external configuration file. These configuration files are generated by the **Protect your application** wizard.

The QLM License Wizard Control is also available as a standalone executable that can run alongside your application.

When you install QLM on your system, a Quick License Manager tab is added to your Visual Studio toolbox that contains the QLM .NET Controls. If for any reason the Quick License Manager tab was not added to your Visual Studio toolbox, you can attempt to recreate this section by clicking on the Refresh button under Options / Enable Visual Studio Integration. Note that the QLM tab is not added to the Visual Studio Express edition as this edition does not support programmatic additions to its toolbox.

For more details about the QLM .NET Controls, refer to the API Reference.

For WPF applications, you can host the QLM Windows Forms Controls in WPF as described in this [article](#).

For validating the license key on subsequent launches of your application, use the **Protect your application** wizard to generate a helper class and add this class to your application. The helper class has a method called

ValidateLicenseAtStartup. You should call this method when your application is launched. For more details about this method, refer to the API reference section.

We also provide several samples in a multitude of programming languages. To locate the sample that matches your application, launch the [Get Started Wizard](#) and follow the instructions to locate the proper sample.

Online Activation using the QLM .NET Basic Activation Control

QLM provides a control that can be dropped in your application to simplify the process of online activation. When using the control, there is almost no need to write any code to implement online activation.

The control is available for applications developed using Microsoft .NET 2.0 or later.

A sample program can be found in the following folder: C:\Users\Public\Documents\Quick License Manager\Samples\qlmpro\Windows\DotNet\C#\QlmControlSample.

To use the QLM .NET Control:

-
- Create a form in your application.
-
- Locate the QLM .NET Controls in the Visual Studio toolbox.
-
- Drag the QlmWebBasicActivationControl and drop it onto your form.
-
- Add a reference to QlmLicenseLib.dll.
-
- In your Visual Studio project, create two new folders: x86 and x64.
-
- In the x86 folder, click Add Existing Item and select redistrib\x86\IsLicense50.dll then set the "Copy To Output" property to "Copy If Newer".
-
- In the x64 folder, click Add Existing Item and select redistrib\x64\IsLicense50.dll then set the "Copy To Output" property to "Copy If Newer".
-
- Locate all the QLM properties (prefixed with Qlm) and update the properties as needed. The properties that must be updated to protect your product are: QlmProductID, QlmMajorVersion, QlmMinorVersion, QlmCommunicationEncryptionKey, QlmPublicKey and QlmWebServiceUrl.

The QlmWebBasicActivationControl also exposes 2 events:

-
- QlmClose is triggered when the Close button is clicked
-
- QlmActivate is triggered when the Activate button is clicked

For more details about all the properties exposed by the .NET Control, review the .NET Control Reference section in the Help.

In addition to integrating the control, you need to modify your application to validate a license key at startup. Once the user has activated his license, the control stores the license key in a hidden location on the end user system. At startup, you should validate that the stored license key is still valid. Use the QLM Code Generator to generate the LicenseValidator class. This class has a method called ValidateLicenseAtStartup which you can call when your application is launched:

```
LicenseValidator lv = new LicenseValidator ();
```

```
if (lv.ValidateLicenseAtStartup(computerID, ref needsActivation, ref returnMsg) == false)
```

```
{
```

```
// the stored license key is not valid. Display the Qlm Control to allow the user to enter a new key
```

Implement Online Activation using the QLM License Wizard .NET Control

QLM provides a control that can be dropped in your application to simplify the process of online activation. When using the control, there is almost no need to write any code to implement online activation.

The control is available for applications developed using Microsoft .NET 2.0 or later.

A sample program QlmLicenseWizardSample can be found in the samples folder.

To use the QLM License Wizard .NET Control:

-
- Create a form in your application.
-
- Locate the QLM .NET Controls in the Visual Studio toolbox.
-
- Drag the QlmLicenseWizardCtrl and drop it onto your form.
-
- Add a reference to QlmLicenseLib.dll.
-
- In your Visual Studio project, create two new folders: x86 and x64.
-
- In the x86 folder, click Add Existing Item and select redistrib\x86\IsLicense50.dll then set the "Copy To Output" property to "Copy If Newer".
-
- In the x64 folder, click Add Existing Item and select redistrib\x64\IsLicense50.dll then set the "Copy To Output" property to "Copy If Newer".
-
- Locate all the QLM properties (prefixed with Qlm) and update the properties as needed. The properties that must be updated to protect your product are:
QlmLicenseProperties.QlmProductID, QlmLicenseProperties.QlmMajorVersion,
QlmLicenseProperties.QlmMinorVersion,
QlmLicenseProperties.QlmCommunicationEncryptionKey,
QlmLicenseProperties.QlmPublicKey QlmLicenseProperties.and QlmWebServiceUrl.

The QlmLicenseWizardCtrl also exposes 3 events:

- QlmClose is triggered when the Close button is clicked
- QlmActivate is triggered when the Activate button is clicked
- QlmCancel is triggered when the Cancel button is clicked

For more details about all the properties exposed by the .NET Control, review the .NET Control Reference section in the Help.

In addition to integrating the control, you need to modify your application to validate a license key at startup. Once the user has activated his license, the control stores the license key in a hidden location on the end user system. At startup, you should validate that the stored license key is still valid. In the QLM Management Console, click the "Protect Your Application" tab and follow the wizard steps to generate the LicenseValidator class. This class has a method called ValidateLicenseAtStartup which you can call when your application is launched:

```
LicenseValidator lv = new LicenseValidator ();  
while (lv.ValidateLicenseAtStartup(computerID, ref needsActivation, ref returnMsg) == false)  
{  
    // the stored license key is not valid. Display the Qlm License Wizard Control to allow the user to enter a  
    new key  
}
```

Online Activation using API calls

To activate a license key, you invoke the `ActivateLicense` or `ActivateLicenseForUser` methods.

`ActivateLicense` should be called when the activation key is already associated to a user.

`ActivateLicenseForUser` should be called to activate the license and associate it to the specified user.

Note that to add a user, you can call the `AddUser` API or call the `ActivateLicenseDialog` which display a license registration form that allows the user to register the license.

Example:

```
QlmLicense lic = new QlmLicense();
lic.CommunicationEncryptionKey = "{B6163D99-F46A-4580-BB42-BF276A507A14}";
lic.DefineProduct(1, "My Product", 1, 1, "encKey",
"{549D9583-7152-41bf-9322-AF0A6DB28223}");
lic.ActivateLicense("http://localhost/qlmservice.asmx", activationKey, computerKey,
computerName, "5.0.00", "my data", out response);
```

```
ILicenseInfo licenseInfo = new LicenseInfo();
string message = string.Empty;
if (target.ParseResults(response, ref licenseInfo, out message))
{
    Console.WriteLine ("PC key=" + licenseInfo.ComputerKey);
}
```

If you need to create an activation key from your web site, use the `CreateActivationKey` method. This method will create an activation key and store it in the database. Note that prior to calling `CreateActivationKey`, you must call the `DefineProduct` function and set the `AdminEncryptionKey` property. In addition, this function can only be called if you update the `web.config` on the web server as follows:

```
<setting name="enableCreateActivationKey" serializeAs="String">
    <value>True</value>
</setting>
```

For security reasons, creating activation keys should always be done from your site and not integrated in your application.

Example:

```
QlmLicense lic = new QlmLicense();
lic.CommunicationEncryptionKey = "{B6163D99-F46A-4580-BB42-BF276A507A14}";
lic.DefineProduct(1, "My Product", 1, 1, "encKey",
"{549D9583-7152-41bf-9322-AF0A6DB28223}");
lic.CreateActivationKey("http://localhost/qlmservice.asmx", "john@sm.com",
255, 1, true, "5.0.00", string.Empty, "my data",
out response);
```

```
ILicenseInfo licenseInfo = new LicenseInfo();
string message = string.Empty;
if (target.ParseResults(response, ref licenseInfo, out message))
{
    Console.WriteLine ("Activation key=" + licenseInfo.ActivationKey);
}
```

How to integrate QLM with your Web Application

There are 2 steps required to protect your application:

- The first time the user launches your application, you need to present the user with a license registration form so that they can enter a license key and activate it.
- On subsequent launches of your application, you read the previously activated license key and validate it before opening up your application.

PROTECTING WINDOWS WEB BASED .NET APPLICATIONS

If you are developing a web based application such as an ASP.NET app or a SharePoint application, the QLM .NET controls cannot be used. A sample program is available that shows how to capture and activate a license key. The sample is located in the following folder:

%Public%\Documents\Quick License Manager\Samples\qlmpro\Windows\DotNet\Basic\C#\vs2008\AspDotNetSample

For validating the license key on subsequent launches of your application, use the **Protect your application** wizard to generate a helper class and add this class to your application. The helper class has a method called **ValidateLicenseAtStartup**. You should call this method when your application is launched. For more details about this method, refer to the API reference section.

Protect Windows 8 Store Apps

QLM Pro can protect Windows 8 Store applications with permanent, trial and device bound keys.

A .NET library (source code included) exposes an API that enables you to provide your customers with an evaluation of your software and then turn it at anytime into a permanent license, or simply extend it for subscription based applications. The .NET library along with a sample applications are provided in the following QLM Pro samples folder:

%Public%\documents\quick license manager\samples\qlmpro\Windows\Win8Store

The sample contains 2 project: Qlm.WindowsStore and Qlm.WindowsStore.Sample

Qlm.WindowsStore

is the library that performs the license validation, activation, decryption, etc. You typically do not need to change any code in this package. The QlmLicense class in this package is the main class you need to interact with. The most common methods of this class are document in the help under "Quick License Manager Professional / API reference / Mobile Devices API".

Qlm.WindowsStore.Sample

simulates your application. When the application is launched, the application attempts to retrieve a stored license on the device to validate it. If no key was ever activated, the user is prompted to enter an Activation Key and activate it.

Use the QLM Pro Application to create an Activation Key from the Manage Keys tab. Note that activation keys can be created from your server using our API or directly from one of the ecommerce provider integrated with QLM. For a full list of ecommerce providers integrated with QLM, check the help or our web site.

Once the user enters the Activation Key, you call the QlmLicense.ActivateLicense method. If activation is successful, digitally signed license information is stored on the device. QLM uses RSA asymmetric encryption to store license information on the device. The RSA public/private key pair is automatically generated by QLM when you define a product in the QLM Application Define Products page. The keys are displayed on the Encryption Keys tab / Mobile Devices Encryption.

Note that the encrypted data stored on the device is signed on the QLM server using the RSA private key and verified on the device using the RSA public key.

In the Qlm.WindowsStore application, the RSA public key is stored in a file called QlmPublicKey.xml. In your own application, it is recommended that you hard code the public key in your code rather than store it in an external file.

In the event a client does not have an internet connection to activate a license online, you can perform an **offline activation**

as described below:

- In the QLM application, under the Manage Keys tab, locate and select the license to activate.
- Click on the Activate button.
- Fill in the Computer ID field on the Activation tab along with other fields as required.
- Click on the **Mobile Device Activation** tab.
- Select a location where you would like to store a license file then click Ok.
- Send the generated license file to your customer and ask them to copy it to the folder where your application expects the license file to be located.

Protect Windows Phone Apps

QLM Pro can protect Windows Phone 7 and Windows Phone 8 applications with permanent, trial and device bound keys.

A .NET library (source code included) exposes an API that enables you to provide your customers with an evaluation of your software and then turn it at anytime into a permanent license, or simply extend it for subscription based applications. The .NET library along with a sample applications are provided in the following QLM Pro samples folder:

```
%Public%\documents\quick license  
manager\samples\qlmpro\Windows\WindowsPhone\WinPhone7  
%Public%\documents\quick license  
manager\samples\qlmpro\Windows\WindowsPhone\WinPhone8  
The sample contains 2 project: Qlm.WPx and Qlm.WPx.Application
```

Qlm.WPx

is the library that performs the license validation, activation, decryption, etc. You typically do not need to change any code in this package. The QlmLicense class in this package is the main class you need to interact with. The most common methods of this class are document in the help under "Quick License Manager Professional / API reference / Mobile Devices API".

Qlm.WPx.Application

simulates your application. When the application is launched, the application attempts to retrieve a stored license on the device to validate it. If no key was ever activated, the user is prompted to enter an Activation Key and activate it.

Use the QLM Pro Application to create an Activation Key from the Manage Keys tab. Note that activation keys can be created from your server using our API or directly from one of the ecommerce provider integrated with QLM. For a full list of ecommerce providers integrated with QLM, check the help or our web site.

Once the user enters the Activation Key, you call the QlmLicense.ActivateLicense method. If activation is successful, digitally signed license information is stored on the device. QLM uses RSA asymmetric encryption to store license information on the device. The RSA public/private key pair is automatically generated by QLM when you define a product in the QLM Application Define Products page. The keys are displayed on the Encryption Keys tab / Mobile Devices Encryption.

Note that the encrypted data stored on the device is signed on the QLM server using the RSA private key and verified on the device using the RSA public key.

In the Qlm.WPx.Application application, the RSA public key is stored in a file called QlmPublicKey.xml. In your own application, it is recommended that you hard code the public key in your code rather than store it in an external file.

In the event a client does not have an internet connection to activate a license online, you can perform an **offline activation**

as described below:

- In the QLM application, under the Manage Keys tab, locate and select the license to activate.
- Click on the Activate button.
- Fill in the Computer ID field on the Activation tab along with other fields as required.
- Click on the **Mobile Device Activation** tab.
- Select a location where you would like to store a license file then click Ok.
- Send the generated license file to your customer and ask them to copy it to the folder where your application expects the license file to be located.

How to integrate QLM with your non .NET Windows application

For non .NET applications, you have 2 options to capture and activate a license:

- Use the standalone *QlmLicenseWizard.exe* application to capture and activate a license
- Create your own license registration form.

The typical command line interface to launch the QLM License Wizard is:

```
QlmLicenseWizard.exe /settings "<path>\settings.xml"  
/uiSettings "<path>\uiSettings.xml"
```

where both xml files referenced above are generated by the [Protect your application](#) wizard.

When the *QlmLicenseWizard.exe* application exits, you should once again call **ValidateLicenseAtStartup** and confirm that the license is valid. If it is not, you either exit your application or launch the *QlmLicenseWizard.exe* application again.

For VB, C++ or any other language that supports interfacing with COM/ActiveX controls, the **Protect your application** wizard generates a helper class that you need to add to your application. The helper class has a method called `ValidateLicenseAtStartup`. You should call this method when your application is launched. If the call to **ValidateLicenseAtStartup** fails or returns that activation is needed, you should then invoke the *QlmLicenseWizard.exe* as described earlier.

We also provide several samples in a multitude of programming languages. To locate the sample that matches your application, launch the [Get Started Wizard](#) and follow the instructions to locate the proper sample.

Online Activation using the QLM License Wizard Standalone Application

QLM provides a standalone application (QlmLicenseWizard.exe) that you can invoke from your application to simplify the process of online activation. When using QlmLicenseWizard.exe, there is almost no need to write any code to implement online activation. This method is ideal for non .NET applications that cannot use the QLM .NET Control. Samples are provided for C++, MS-Access and Excel.

The QlmLicenseWizard.exe can be used on any Windows XP and higher system and requires Microsoft .NET 2.0 or later.

To use the QlmLicenseWizard.exe application:

- From the QLM Console, create a product.
- Go to the **Protect your application** tab and follow the instructions in the wizard.
- Modify your application as follows:
 - When your application is launched, call the *ValidateLicenseAtStartup* function. This function is generated by the **Protect your application** wizard in the *LicenseValidator* class.
 - If the ValidateLicenseAtStartup function returns false, launch the QlmLicenseWizard.exe with the /settings argument.
 - When the QlmLicenseWizard.exe process exits, call *ValidateLicenseAtStartup* again until the license is valid.

Once the user has activated his license from the QlmLicenseWizard process, the license key is automatically stored in a hidden location on the end user system.

Example:

```
LicenseValidator lv = new LicenseValidator ();
while (lv.ValidateLicenseAtStartup(computerID, ref needsActivation, ref returnMsg) == false)
{
    // Launch the QlmLicenseWizard.exe process
    // If the process exit code is 4, break the while loop
}
```

The command line arguments of QlmLicenseWizard.exe are:

Argument	Meaning
/settings	Path to the settings file generated by the Protect your application wizard (enclose the full path in double quotes).
/computerID	The unique identifier of the current system. If not argument is provided, the computer name is used.
/activationKey	Optional argument. By default, the activation key is automatically retrieved from the location associated with your products by calling the ReadKeys API. To override this behavior, you can specify the activationKey on the command line.
/computerKey	Optional argument. By default, the computer key is automatically retrieved from the location associated with your products by calling the ReadKeys API. To override this behavior, you can specify the activationKey on the command line.
/floating_master	Optional argument. Displays the floating license page when activating the master node.

/floating_node	Optional argument. Displays the floating license page when activating a client node.
/appversion	Optional argument. Sets the version of the application that's launching the wizard. This version is used in the Check for Updates feature.
/showur	Optional argument. Displays the User Registration page.

Upon exiting, the QlmLicenseWizard.exe process returns the following exit codes:

Exit Code	Meaning
0	the license is valid
1	the license is not valid
2	the license has expired
3	usage error in the command line, typically indicating some missing or invalid arguments
4	the user canceled the wizard
10	the user successfully deactivated the license
11	the user failed to deactivate the license

Online Activation using http requests

To activate a license using an http request, use the following command:

http://yourserver/qlm/qlmservice.asmx/ActivateKey?is_productid=4&is_majorversion=3&is_minorversion=0&is_pcid=00-30-BD-92-74-25&is_avkey=A2CC8-0F116-8EA0A-CC2C10

where is_pcid is the unique identifier for a PC and is_avKey is the value returned from GetActivationKey.

For details about the arguments, review the Help under Quick License Manager Professional / Http Methods / GetActivationKey

To invoke GetActivationKey, you need to send an http request as follows:

http://yourserver/qlm/qlmservice.asmx/GetActivationKey?is_productid=1&is_majorversion=1&is_minorversion=0

where is_productid, is_majorversion and is_minorversion are the values defined for the required product in Quick License Manager.

Basic eCommerce Providers

A basic eCommerce provider is a provider that does not allow calling an external script such as the QLM License Server during the purchase process. This type of provider typically allows you to upload a list of license keys that are distributed to buyers upon purchase. In order to integrate with basic eCommerce providers, follow the steps below:

- Use QLM to generate a set of activation keys that are not bound to any user.
- To create these keys, click on Licenses, and select Create.
- Select your product and the number of license keys to create
- Uncheck the Customer Email checkbox and the Single Activation Key checkbox.
- Specify the Number of Licenses to create
- In the Results dialog, double click on the result.
- Copy the generated keys to the clipboard and paste them in your eCommerce provider's web site.

In your application, you now need to enter the user information and activate the license key. The QLM License Wizard includes a User Registration page to capture user information and publish it to the QLM License Server. To enable the User Registration page, set the `QlmShowUserRegistrationPage` property to true when configuring the QLM License Wizard properties in the Protect Your Application wizard. Alternatively, you may create your own form to capture user information. If you do so, you will need to call the following functions:

- `DefineProduct`
- `AddUser`
- `ActivateLicenseForUser`

Ecommerce Providers

If you are using an ecommerce provider to sell your software, QLM provides a mechanism for integrating with your ecommerce provider. The QLM License Server provides a method that can be invoked from your ecommerce provider as follows:

http://yourserver/yourvirtualdirectory/qlmservice.asmx/GetActivationKey?is_productid=<productID>&is_majorversion=<majorVersion>&is_minorversion=<minorVersion>&is_vendor=<vendor>&is_features=<features>

where,

<productID> is the 2 digit product identifier

<majorVersion> is the major version of your product

<minorVersion> is the minor version of your product

<vendor> is the name of the ecommerce provider as specified in Manage Keys / 3rd Party Extensions

<features>

are the features to enable in this key

The following additional arguments can also be specified:

<is_usemultipleactivationskey> if true, a single activation key is return for a multiple license request.

<is_userdata1> user data to associate to the license key

<is_affiliateid> affiliate id to associate to the license key

When the GetActivationKey method is invoked, the License Server issues an activation key based on the provided url arguments (is_productid, is_majorversion, is_minorversion, is_vendor, is_features).

Since each ecommerce provider uses a different format for its input (the set of fields entered by the customer on the order form), and expects results in their own format, a custom solution is required for each provider.

Authentication of the call to GetActivationKey is performed by matching the credentials configured in the QLM / 3rd Party Extensions against the credentials sent by the ecommerce provider as part of the URL arguments.

Note that most the methods exposed by the License Server cannot be called with a URL except for a few methods such as GetActivationKey and ActivateKey. The full list of these methods is documented in the API reference / License Server Http methods section. All other web methods implement a secure authentication mechanism that only accepts requests from the QLM Management Console or the QLM API.

Example:

http://www.mydomain.com/qlm/qlmservice.asmx/GetActivationKey?is_productid=2&is_majorversion=3&is_minorversion=0&vendor=digibuy&is_features=3

The URL above generates an activation key for ProductID = 2, MajorVersion = 3, MinorVersion = 0.

The returned response is customized for Digibuy (e-commerce provider).

The enabled features are: Feature 1 and Feature 2 (1+2=3).

Avangate

If you are using Avangate as an ecommerce provider, QLM integrates seamlessly with Avangate's ordering system. After completing the steps below, when a customer purchases your product from Avangate, Avangate will automatically get a license key from the QLM License Server and then update the QLM database with the license and customer information.

To have Avangate invoke QLM during an order, do the following in Avangate Control Panel:

-
- Click on the Setup menu item.
-
- Click on "Configure electronic delivery settings".
-
- In the "Add New Code List" panel, enter a List Name, set the List type to "Dynamic" and click on "Add List".
-
- Click on the "Debug ..." button next to the URL field.
-
- In the Debug Url field enter the following value:
 - o https://quicklicensemanager.com/solvusoft/qlm/qlmservice.aspx/GetActivationKey?is_vendor=Avangate&is_productid=1&is_majorversion=1&is_minorversion=0&is_qlmversion=5.0.00;
 - o You should update the URL above to point to your server and specify your product ID, major and minor version.
 - o Select any product from the list.
 - o Click on the "Step 3. Debug" button.
 - o When you get the "Success" message, click on the "Save" button.
 - o Close the dialog.
 - o Click on the "Update" button

Finally, in the Assigned products section, add the required product to the "Assigned Products" list.

Note that for security reasons, you can also specify a user name / password in the URL above.

For example:

https://quicklicensemanager.com/solvusoft/qlm/qlmservice.aspx/GetActivationKey?is_vendor=avangate&is_productid=1&is_majorversion=1&is_minorversion=0&is_qlmversion=5.0.00&is_user=john&is_pwd=12345;

The same user/password combination must be specified in the QLM Application under Manage Keys / Commerce Providers for the Avangate entry.

Now that the Electronic delivery list is created, you need to associate it to your product. To do so:

- Click on the Products menu item then select your product.
- Click on the Fulfillment tab.
- Select "Fulfillment made through Avangate delivery (binary keys, activation codes, Backup Media, product file, DIS)"
- In the Content & methods section, select "Electronic code / key / binary file";
- In the "Activation codes settings" section, select the Code list you created earlier.
- In the "Additional fulfillment information - by email", customize the email template.

To place a test order:

- Click on the Products menu item then select your product.
- Click on the Information tab.
- Click on "Get buy links for this product"
- Scroll down the page and click on "Place a test order"
- In the "Activation codes settings" section, select the Code list you created earlier.
- In the "Additional fulfillment information - by email", customize the email template.

Cleverbridge

If you are using Cleverbridge as an eCommerce provider, QLM integrates seamlessly with Cleverbridge's ordering system.

After completing the steps below, when a customer purchases your product from Cleverbridge, Cleverbridge will automatically get a license key from the QLM License Server service and then update the QLM database with the license and customer information.

To integrate Cleverbridge with QLM, perform the following in the Cleverbridge Control Panel:

Cleverbridge Setup for Protecting your product

-
- In the Cleverbridge Commerce Assistant, go to Products and Delivery then select Key Generators
-
- Click "Add Key Generator / Web Key Generator"
-
- In the Name field, type: QLM Key Generator
-
- In the Path field, type:
https://qlm3.net/qlmdemov16/QlmLicenseServer/qlmservice.asmx/GetActivationKeyWithExpiryDate?is_vendor=cleverbridge&is_productid=1&is_majorversion=1&is_minorversion=0&is_qlmversion=5.0.00
-
- You should update the URL above to point to your server and specify your product ID, major and minor version.
-
- Leave the "Authenticate" checkbox unchecked. Note that you can add a user/pwd to the URL above. See the QLM help for more details.
-
- In the Interface field, select "Type 1"
-
- In the Character Encoding field, select "Unicode (UTF-8)"
-
- Check "Use Romanized contact values"
-
- Uncheck "Call once per purchase"
-
- Check "Use XML client notification"
-
- In the XML Schema field, select "Use Current Version"
-
- Uncheck "Client handles errors"
-
- Click Save
-

- In the Products section, locate your product and double click it.
-
- In the Delivery Details section, set the "Delivery type" to "company delivers key"
-
- In the Web Key Generator field, select "QLM Key Generator"

With the steps above completed, place a test order as follows:

-
- Click "Tools / Link Generator"
-
- Set the Destination field to "Checkout process"
-
- In the Cart Content section, select your product and click on the "Down" key"
-
- Click Open and select a browser
-
- Fill in all the required fields and place a test order.
-
- When the order is completed, a license key will be generated and an email will be sent to the customer with the license key. Additionally, customer information will be transferred to the QLM database.

Digibuy

If you are using Digibuy as an ecommerce provider, QLM integrates seamlessly with Digibuy's ordering system.

To have Digibuy invoke QLM during an order, do the following in Digibuy Admin Console:

-
- Select a Product.
-
- Click on "reg codes".
-
- Locate the **http://** field
-
- Enter the following URL:

http://yourserver/yourvirtualdirectory/qlmservice.asmx/GetActivationKey?is_productid=<productID>&is_majorversion=<majorVersion>&is_minorversion=<minorVersion>&is_vendor=digibuy&is_features=<features>where

- [is_vendor = Digibuy](#)
- [is_productid = your product id as defined in QLM](#)
- [is_majorversion = your product's major version as defined in QLM](#)
- [is_minorversion = your product's minor version as defined in QLM](#)
- [is_qlmversion = 5.0.00](#)
- [is_features = semi comma separated list of feature sets and their corresponding values. Example: is_features=0:3;1:1. This means that in feature set 0, features 1 + 2 are enabled and in feature set 1, feature 1 is enabled.](#)
- [is_user = username defined in Manage Keys / 3rd Party Extensions \(optional\)](#)
- [is_pwd = password defined in Manage Keys / 3rd Party Extensions \(optional\)](#)

[Additionally, Digibuy provides a mechanism for protecting this request with a password. On the same page as above, enter a password in the **Password** field.](#)

[The password must also be updated in the QlmProviders.xml file located in the bin folder. Edit QlmProviders.xml using any text editor and update the user and password fields.](#)

[In the user field, specify your Digibuy user id. In the password field, specify the password entered above.](#)

```
<provider>
  <name>Digibuy</name>
  <class>Qlmsvc.Digibuy</class>
  <dll>QlmDigibuy.dll</dll>
  <user>
  </user>
  <password>
  </password>
  <passwordEncrypted>>false</passwordEncrypted>
</provider>
```

With the steps above completed, place a test order. When the order is placed, the following will occur:

-
- A new user will be automatically added to the QLM database based on the information collected during the ordering process.
-
- An activation key will be created in the QLM database and associated to the user.
-
- An email will be sent to the customer with the activation key.

FastSpring

If you are using FastSpring as an eCommerce provider, QLM integrates seamlessly with FastSpring's ordering system.

After completing the steps below, when a customer purchases your product from FastSpring, FastSpring will automatically get a license key from the QLM License Server and then update the QLM database with the license and customer information.

To integrate FastSpring with QLM, perform the following in the FastSpring Control Panel:

FastSpring Setup for a Basic Product

-
- Select your product in the FastSpring Control Panel
-
- Add a Fulfillment Action:
-
- On the Licenses tab, select "Remote" and click on Next
-
- Set the URL to:
https://qlm3.net/qlmdemov16/QlmLicenseServer/qlmservice.asmx/GetActivationKey?is_vendor=fastspring&is_productid=1&is_majorversion=1&is_minorversion=0&is_qlmversion=5.0.00
-
- You should update the URL above to point to your server and specify your product ID, major and minor version.
-
- Method: HTTP POST
-
- POST Encoding: UTF-8
-
- License Name Type: Person Full Name + Email Address
-
- Output Format: Plain Text, Single-Line License
-
- Click on Create
-
- Click on Save
-
- In the Fulfillment Actions section, click on Add
-
- Select Email / Web Notification and click Next
-
- Reuse Options: Reusable On Multiple Products
-
- Subject : #{orderItem.display} - Order #{order.reference}
-
- Email Text Contents :
- Dear #{order.customer.fullName},

- Thank you for your purchase of #{orderItem.display}.
- Your Activation Key is: #{orderItem.fulfillment.license.outcome.licenses.list}
- You can download #{orderItem.display} from:
- <http://soraco.co/products/qlm/qlmsetup9.exe>
- Should you have any questions please contact us at: sales@soraco.co
- Sincerely,
- Soraco Technologies
-
- You can customize the Email Html and Web tabs as well.

FastSpring Setup for Updating Contact Information

To transfer the contact information from FastSpring to QLM, follow the steps below:

-
- From the FastSpring Control Panel Home page, click on the "Notify" icon in the top toolbar.
-
- Click on Add Notification Rule.
-
- Format: HTTP Remote Server Call.
-
- Type: Order Notification.
-
- Remote Server URL:
<https://qlm3.net/qlmdemov16/QlmLicenseServer/qlmservice.asmx/UpdateUserInformation>
-
- Click on Next
-
- In the HTTP Parameters section, click on Add Parameter.
-
- Name: is_avkey
-
- Value: #{orderItem.fulfillment.license.outcome.licenses.list}
-
- Click on Add Parameter again.
-
- Name: is_vendor
-
- Value: fastspring
-
- Optionally, add an is_user and is_pwd parameters to enforce authentication. The user and password can be specified from the QLM Application, under Manager Keys / Commerce Providers.

With the steps above completed, place a test order by clicking on the FastSpring Control Panel Home page, then Store Testing. Once your test order is completed, an activation key will be created in the QLM database along with the corresponding customer and order information.

FastSpring Setup for Maintenance Plan

If you sell a yearly maintenance plan to your customer, you can configure FastSpring to automatically update the maintenance plan expiry date in QLM.

-
- Click on Products and Pages
-
- Click on Create Subscription Product
-
- Name: Yearly Maintenance Plan
-
- Regular Period Length: 1 year
-
- USD: xx
-
- Click on Create
-
- In the Fulfillment section, click on Add
-
- Click on the License tab, select Remote and click Next
-
- URL:
https://qlm3.net/qlmdemov16/QlmLicenseServer/qlmservice.aspx/RenewMaintenancePlan?is_vendor=fastspring&is_user=john&is_pwd=fast123
-
- License Name Type: Person Full Name + Email
-
- Click on Create
-
- Click on Save
-
- Click on the Home button
-
- Click on Custom Fields
-
- Click on Create Customer Field Configuration
-
- Name: custom_referrer (THIS NAME IS FIXED, IT CANNOT BE CHANGED)
-
- Click on Next
-
- Display: Activation Key
-
- Click on Add Form Field
-
- Input Type: Textbox
-
- Required: Yes

-
- Name: custom_referrer
-
- Question Text: Activation Key
-
- Question Description: Enter your current activation key (starts with the letter A)
-
- Click on Add
-
- Click on Save
-
- Click on Save
-
- In the Conditions section, click on Edit
-
- Select Order Environment Condition and click Next
-
- Environment Tag Exists: is_maintenance_plan
-
- Click on Create
-
- Click on Save
-
- Click on the Home page

To test the maintenance plan configuration:

-
- Click on Store Testing
-
- Click on Optional Parameters
-
- Tags: is_maintenance_plan
-
- Click on Testing Links tab then Detail for “demo yearly maintenance plan?o:p>
-
- Click on Order Now, fill contact info then Next
-
- The next page should show the Activation Key field.
-
- Go to the QLM Application and locate the license created above.
-
- Enter the Activation Key in the order form and click on Order
-
- Once the order is completed, go to QLM and locate your license key
-
- Refresh the page, select the license and click on the Edit.
-
- Note how the maintenance plan was increased by a year.

FastSpring Setup for Purchasing an Upgrade

If you sell upgrades to a specific version of your software, you can integrate FastSpring with QLM's license upgrade feature.

-
- Create a Product in a way similar to the first section
-
- Click on Create Product
-
- Name: Upgrade
-
- USD: xx
-
- Click on Create
-
- In the Fulfillment section, click on Add
-
- Click on the License tab, select Remote and click Next
-
- URL:
https://qlm3.net/qlmdemov16/QlmLicenseServer/qlmservice.aspx/UpgradeLicense?is_vendor=fastspring&is_productid=1&is_majorversion=2&is_minorversion=0&is_qlmversion=5.0.00&is_user=john&is_pwd=fast123
-
- License Name Type: Person Full Name + Email
-
- Click on Create
-
- Click on Save
-
- Create an Email Fulfillment similar to the main Product
-
- Click on the Home button
-
- Click on Custom Fields
-
- Click on "custom_referrer?o:p>
-
- In the Conditions section, click on Edit
-
- In the Active Conditions section, click Edit
-
- Environment Tag Exists: is_maintenance_plan, is_upgrade
-
- Click on Save
-
- Click on the Home page

To test the Product Upgrade configuration:

-
- Click on Store Testing
-
- Option Paramaters / Tags: is_upgrade
-
- Click on testing links: Demo Upgrade / Detail
-
- Click on Order Now, then Next
-
- In the QLM Application, create a license key for product Demo 1.0
-
- Enter this activation key in the FastSpring form
-
- Click on Complete Order
-
- Once the order is completed, go to QLM and click on Today's orders
-
- Note how a new key is created replacing the old one.

FastSpring Setup for a Subscription Product

If you sell a subscription based product, you can integrate FastSpring with QLM's subscription renewal feature.

-
- Create a Subscription based Product in FastSpring
-
- Click on Create Subscription Product
-
- Name: Your Product
-
- USD: xx
-
- Click on Create
-
- In the Fulfillment section, click on Add
-
- Click on the License tab, select Remote and click Next
-
- URL:
https://qlm3.net/qlmdemov16/QlmLicenseServer/qlmservice.asmx/GetActivationKeyWithExpiryDate?is_vendor=fastspring&is_productid=1&is_majorversion=1&is_minorversion=0&is_qlmversion=5.0.00&is_user=john&is_pwd=fast123&is_expduration=365
- 5
-
- Method: HTTP POST
-
- POST Encoding: UTF-8
-
- Output Format: Single-Line License

-
- Go to the Advanced tab
-
- Set the Fulfillment Applicability to: Applies to Non-Rebills / First Orders Only
-
- Go back to the Fullfillment section and click on Add
-
- Click on the License tab, select Remote and click Next
-
- URL: https://qlm3.net/qlmdemov16/QlmLicenseServer/qlmservice.aspx/RenewSubscriptionHttp?is_vendor=fastspring&is_user=john&is_pwd=fast123&is_expduration=365
-
- Method: HTTP POST
-
- POST Encoding: UTF-8
-
- Output Format: Single-Line License
-
- Go to the Advanced tab
-
- Set the Fulfillment Applicability to: Applies Rebills Only
-
- Create an Email Fulfillment similar to the main Product (details as described in the previous sections).
-
- Click on Save

To test the Product Upgrade configuration:

-
- Click on Store Testing
-
- Select your subscription based product and click on Detail
-
- Go through the order process
-
- A subscription based license key will be created.
-
- In the QLM Application, verify that the license key was indeed created.
-
- In FastSpring, click on the Reports button in the toolbar
-
- Next to the Recent Orders button, click on View
-
- Locate the order you just placed and click on arrow button.
-
- In the Subscriptions panel, click on the product name
-
- In the Actions panel, click on Simuate Next Billing

-
- In the QLM Application, refresh the grid view and verify the the SubscriptionExpiryDate was pushed by a year.

Google

If you have a merchant account with Google Checkout and are using Google Checkout as your ecommerce provider, QLM can issue activation keys automatically from Google Checkout's ordering system.

In order to generate license keys for an order and send the keys to a customer automatically at the time of purchase, QLM can be used in conjunction with Google's Checkout Merchant Account Settings. Google Checkout provides a mechanism for immediate notification when a customer purchases your product. QLM integrates with this mechanism using **QlmGoogleCheckout**. **QlmGoogleCheckout** is an ASP.Net application included with QLM.

When an order is charged either manually by the merchant or automatically by Google Checkout, **QlmGoogleCheckout** adds the customer to the QLM database, creates an order in QLM, generates one or more activation keys, and emails the activation keys to both the merchant and the customer. An order can include multiple items. One activation key per item is issued. Each item must correspond to a product defined in QLM.

The Merchant's Shopping Cart

The following must be added to your shopping cart to map an item to its corresponding QLM product. If using an XML shopping cart add the following to your cart for each item:

```
<merchant-private-item-data>productid=1 major=1 minor=0</merchant-private-item-data>
```

If using an HTML shopping cart add the following to your cart for each item:

```
<input type="hidden" name="shopping-cart.items.item-1.merchant-private-item-data" value="productid=1 major=1 minor=0" />
```

- Where productid represents your QLM productID
- Where major represents the major version of your QLM product
- Where minor represents the minor version of your QLM product
- Note that including a second item in an HTML cart would read **shopping-cart.items.item-2....**

The Merchant's Google Checkout Account Settings

The following steps are required to integrate QLM with Google Checkout

- Set up the QLM License Server on a web server by following installation steps found in the QLM Professional Help.
- Define and upload your products to the QLM License Server. Refer to the QLM Help for further information.
- Set up the QLM ASP.Net application **QlmGoogleCheckout** into its own virtual directory. The virtual directory must be configured to require SLL or TLS.
- Configure the **QlmGoogleCheckout** application's Web.config file. Set the following application settings fields:
 - <appSettings>
 - <add key="company" value="Your Company Name"/>
 - <add key="supportEmail" value="support@yourdomain.com"/>
 - <add key="smtpServer" value="localhost"/>
 - <add key="smtpPort" value="25"/>
 - <add key="communicationEncryptionKey" value="communicationKeyUsedByQLMWeb"/>
 - <add key="qlmVersion" value="5.0.00"/>
 - <add key="webServiceUrl" value="http://yourserver/qlmweb/qlmservice.asmx"/>
 - <add key="merchantID" value="Your Merchant ID"/>
 - <add key="merchantKey" value="Your Merchant Key"/>
 - </appSettings>
- From your Google Checkout Merchant account, set the following settings

- From the Settings Tab, select the Integration link on the left hand side.
- Set the API callback URL to the program to QLMGoogleCheckout's GoogleNotify.aspx
The URL might look something like <http://yourdomain.com/qlmgoogle/GoogleNotify.aspx>
- Set the Callback method to XML
- (Optional) If you wish to capture the buyer's phone number in QLM, select the Advanced Settings and and Check the checkbox next to "Return the buyer's billing phone number in the new order notification.

QLM Order Processing

When a customer purchases your product, Google Checkout sends a new order notification. QLM will at this point create a new order in its own database. If the customer does not already exist, a new customer will be added. The customer's billing coordinates are saved. A QLM order consists of the following fields:

- ActivationKey
- ProductName
- MajorVersion
- MinorVersion
- OrderDate
- ActivationDate
- OrderID
- NumLicenses

Financial information such as the amount of the order is not stored in QLM. When you charge the order from your Google Merchant Account, Google Checkout sends a state change notification. QLM will at this time, send the customer by email his or her activation keys. The merchant is also sent the activation keys by email. All other notifications from Google Checkout are ignored.

Below is a typical example of the email sent by QLM:

Dear customer name,

Please find below additional information regarding your purchase.

Google order number: 17274818052274

Date of purchase: 11/28/2007

Product: your product name1

Quantity: 1

Activation Key: A00F-C858-8FE1-1340-6001-00A0

Product: your product name2

Quantity: 2

Activation Key: A0B5-C078-97FA-1220-4102-00A0

For additional inquiries, please contact support@yourdomain.com with the above information.

Thank you,

Your company name

QLM Google Checkout Deployment

To deploy the QLM Google Checkout module to your server, follow the steps below:

- Create a virtual directory on your IIS server called qlmgoogle.

- Configure the virtual directory to require an SSL connection.
- Copy all the files located in the %Public%\Public Documents\Quick License Manager\DeployToServer\QlmGoogleCheckout folder to the qlmgoogle virtual directory.
- Edit the web.config and customize it as described above.

Paypal

If you are using Paypal as an ecommerce provider, QLM integrates seamlessly with Paypal's ordering system. In order to generate a license key for an order and send the key to a customer automatically at the time of purchase, QLM can be used in conjunction with PayPal's Instance Payment Notification (IPN). PayPal's IPN provides immediate notification when a customer purchases your product.

QLM's integration with Paypal performs the following tasks:

1. It validates the paypal request.
2. It contacts the QLM License Server and issues an activation.
3. It sends an email to the customer with the activation key.

In order to set up the IPN, you need to modify your PayPal account's Profile to enable Instant Payment Notification and specify a URL to the QLM service that handles Paypal Notifications. Alternatively, you can activate IPN by including the `notify_url` in your PayPal button HTML.

You can also use the [IPN Simulator](#)

to test the integration.

The URL to the QLM service that handles Paypal notifications is:

`http://yourserver/qlm/qlmpaypalipn.aspx`

For detail instructions on Instant Payment Notification, please refer to PayPal's Order Management Integration Guide found on its website under the tab Merchant Services.

Process Customization

The QLM Site Server Properties enable you to customize all Paypal specific properties. You can access the Server Properties from the Manage Keys tab / Sites / Server Properties.

Description of Paypal Settings:

- `vendorCompanyName`: Name of your company.
- `vendorCompanyEmail`: Your email address. QLM will cc you every time an email is sent to a customer.
- `tempalteFile`: Name of the template file. The template file contains the body of the email message that is sent to the customer. The template contains variables that will be replaced at runtime. The template file is located in the same folder as `qlmpaypalipn.aspx`.
- `smtpServer`: The SMTP server to use when sending emails.
- `smtpUser`: The credentials of the user that needs to authenticate with the SMTP server.
- `smtpPassword`: The password of the user that needs to authenticate with the SMTP server.
- `smtpPort`: The port of the SMTP server.
- `paypalUrl`: Url of the paypal service. When testing in a sandbox, the URL is typically: <https://ipnpb.sandbox.paypal.com/cgi-bin/webscr>. When going live, change the URL to: <https://ipnpb.paypal.com/cgi-bin/webscr>
- `qlmWebServiceUrl` = URL to the QLM License Server, typically located in the same virtual directory.
- `defaultUrlArgs`=When configuring your paypal cart or Buy Now page, you typically need to configure the `productid`, major version and minor version (more details later). If these arguments are not provided, the system will default to the product, major and minor version defined in this setting.
- `loggingLevel`: specifies the level of logs generated. The levels are defined as follows: Errors: 1 - Warnings: 2 - Information: 4 - Verbose: 8. To log errors and warnings, set the `loggingLevel` to 3. To log all events, set the `loggingLevel` to 15.
- `ignoreCustomArgument`: When set to true, the paypal **custom** argument is ignored by QLM. The default is false.
- `ignoreItemNumberArgument`: When set to true, the paypal **item_number** argument is ignored by QLM. The default is false.

- **paypalFields:** PayPal posts variables to the IPN process. Some of these variables are declared in your shopping cart or in your company's purchase page and others are sent by PayPal automatically. For example, your purchase page defines your company by using the variable called "business". Similarly, PayPal uses the variable `payer_email` to identify the customer's email address. The `paypalFields` setting lists the paypal fields that QLM will process. A list of the most common fields is preconfigured but you can add more fields if needed. Any field that is added can be used in the email template.

When configuring your Buy Now button or your Paypal cart, you must configure the following IPN variables. These variables are required by the `QlmPaypalIPN` process:

- **item_name:** product's name, must be changed to be the name of your product.
- **item_number** or **custom:**
`&is_productid=x&is_majorversion=y&is_minorversion=z&is_features=0:1;1:3`
 where x,y,z and f must be replaced with the values that correspond to your product.

When working with carts, Paypal appends a digit at the end of **item_name** and **item_number** for each product listed in the cart. QLM will process all products in the cart and send a single email to the customer with the license keys for all the selected products. Additionally, you can customize the email template to use when sending an email to the customer. To customize the email template, add the `is_emailtemplate` argument as follows: `&is_emailtemplate=template1.txt`. The email template files must be located in the License Server folder, in the same location as the default `QlmEmailTemplate.txt` template file.

The email template can contain any paypal variable such as `%payer_email%` or `%ProductName%`.

When your cart includes multiple items, each item will be listed in the email as defined in the `%ItemTemplate%` section. The `%ItemTemplate%` section starts with an `%ItemTemplate%` tag and end with a corresponding tag. All lines in between the 2 tags are repeated for each item in the cart.

Testing

Once you have completed the customization above, you can use the Paypal IPN simulator to test that the ordering process is working as expected.

The Paypal IPN simulator is located here:

https://developer.paypal.com/webapps/developer/applications/ipn_simulator

In the IPN simulator, set the following fields:

- IPN handler URL: <http://yourserver/qlm/qlmpaypalipn.aspx>
- Transaction type: eCheck Complete
- Payment type: eCheck
- `payment_status`: Completed
- Enter the buyers information
- `custom`: `&is_productid=x&is_majorversion=y&is_minorversion=z&is_features=0:1;1:3` (update these values to match your product)

Plimus

If you are using Plimus as an ecommerce provider, QLM integrates seamlessly with Plimus's ordering system.

To have Plimus invoke QLM during an order, do the following in Plimus Control Panel:

-
- Click on My Account
-
- Locate your Product and click on Setup.
-
- Locate a contract and click on Setup.
-
- Click on the **License Keys** link.
-
- In the License Group/Method field, select: **Custom HTTP request**.
-
- In the Call Method field, select: **One call per Order**.
-
- In the URL for HTTP request, enter the following:
`http://yourserver/qlm/qlmservice.asmx/GetActivationKey?is_vendor=plimus&is_productid=1&is_majorversion=1&is_minorversion=0&is_qlmversion=5.0.00&is_features=0&order_id=<INVOICE_ID>&email=<CUSTOMER_EMAIL>&name=<CUSTOMER_NAME>&company=<COMPANY_NAME>&addr1=<CUSTOMER_ADDRESS1>&addr2=<CUSTOMER_ADDRESS2>&city=<CUSTOMER_CITY>&state=<CUSTOMER_STATE>&zip=<CUSTOMER_ZIPCODE>&country=<CUSTOMER_COUNTRY>&phone=<CUSTOMER_PHONE>&quantity=<QUANTITY>`

where

- is_vendor = plimus
- is_productid = your product id as defined in QLM
- is_majorversion = your product's major version as defined in QLM
- is_minorversion = your product's minor version as defined in QLM
- is_qlmversion = 5.0.00
- is_features = semi comma separated list of feature sets and their corresponding values. Example: is_features=0:3;1:1. This means that in feature set 0, features 1 + 2 are enabled and in feature set 1, feature 1 is enabled.
- is_user = username defined in Manage Keys / 3rd Party Extensions (optional)
- is_pwd = password defined in Manage Keys / 3rd Party Extensions (optional)

With the steps above completed, place a test order. When the order is placed, the following will occur:

-
- A new user will be automatically added to the QLM database based on the information collected during the ordering process.
-
- An activation key will be created in the QLM database and associated to the user.
-
- If you would like to customize the email that is sent to the customer with the activation key, in the Plimus control panel, click on the **Order Email** link and add the following line to the body of the email:

Your Activation Key: <LICENSE_KEYS>

Regnow

If you are using Regnow as an ecommerce provider, QLM integrates seamlessly with Regnow's ordering system.

To have Regnow invoke QLM during an order, do the following in Regnow Control Panel:

-
- Click on Products
-
- Select a Product and click on Edit.
-
- Click on the **Manage Delivery Options** tab.
-
- Locate the **Method** field and select **Post**.
-
- Locate the **http://** field and enter the following URL:

http://yourserver/yourvirtualdirectory/qlmservice.aspx/GetActivationKey?is_productid=<productID>&is_majorversion=<majorVersion>&is_minorversion=<minorVersion>&is_vendor=regnow&is_features=<features>

where

- is_vendor = regnow
- is_productid = your product id as defined in QLM
- is_majorversion = your product's major version as defined in QLM
- is_minorversion = your product's minor version as defined in QLM
- is_qlmversion = 5.0.00
- is_features = semi comma separated list of feature sets and their corresponding values. Example: is_features=0:3;1:1. This means that in feature set 0, features 1 + 2 are enabled and in feature set 1, feature 1 is enabled.
- is_user = username defined in Manage Keys / 3rd Party Extensions (optional)
- is_pwd = password defined in Manage Keys / 3rd Party Extensions (optional)

With the steps above completed, place a test order. When the order is placed, the following will occur:

-
- A new user will be automatically added to the QLM database based on the information collected during the ordering process.
-
- An activation key will be created in the QLM database and associated to the user.
-
- An email will be sent to the customer with the activation key.

ShareIt

If you are using ShareIt as an e-commerce provider, QLM can be used to issue license keys and save the order and customer information when a customer purchases your product using ShareIt's ordering system.

To have ShareIt invoke QLM at the time of purchase, do the following:

-
- Setup the QLM License Server on your server
-
- Optionally, configure authentication to connect to ShareIt. To configure authentication, click on Manage Keys / Tools / eCommerce Providers and specify the credentials for ShareIt.
-
- Create or edit your product in ShareIt's Control Panel
-
- Ask ShareIt personnel to set your product's **Key Generator URL** to the following URL:
`http://yourserver/yourvirtualdirectory/qlmservice.asmx/GetActivationKey?&is_vendor=ShareIt&is_productid=<productid>&is_majorversion=<majorversion>&is_minorversion=<minorversion>&is_qlmversion=<qlmversion>&is_pwd=<pwd>&is_user=<username>&is_features=`
Where:
 - is_vendor = ShareIt
 - is_productid = your product id as defined in QLM
 - is_majorversion = your product's major version as defined in QLM
 - is_minorversion = your product's minor version as defined in QLM
 - is_qlmversion = 5.0.00
 - is_features = semi comma separated list of feature sets and their corresponding values.
Example: is_features=0:3;1:1. This means that in feature set 0, features 1 + 2 are enabled and in feature set 1, feature 1 is enabled.
 - is_user = username defined in Manage Keys / 3rd Party Extensions (optional)
 - is_pwd = password defined in Manage Keys / 3rd Party Extensions (optional)

With the steps above completed, place a test order. When the order is placed, the following will occur:

-
- A new user will be automatically added to the QLM database based on the information collected during the ordering process.
-
- An activation key will be created in the QLM database and associated to the user.
-
- An email will be sent to the customer with the activation key.

SWREG

If you are using SWREG as an ecommerce provider, QLM integrates seamlessly with SWREG's ordering system.

To have SWREG invoke QLM during an order, do the following in the SWREG Control Panel:

-
- Click on Create/Edit Products
-
- Select a Product and click on Edit.
-
- Click on the **Edit Delivery Method**.
-
- Click on **Edit Email**.
-
- Locate the **Keycode generator URL** field and enter the following URL:
<http://yourserver/yourvirtualdirectory/qlmservice.asmx/GetActivationKey>

When invoking the SWREG order form from your web site, add the following argument to the SWREG URL:

`&t=pid%3d<productID>%26mj%3d<majorVersion>%26mn%3d<minorVersion>%26vn%3dSWREG%26fe%3d<features> %26ur%3d<user> %26pw%3d<password> &x=1`

where,

- vn = SWREG
- productID = your product id as defined in QLM
- majorVersion = your product's major version as defined in QLM
- minorVersion = your product's minor version as defined in QLM
- features = semi comma separated list of feature sets and their corresponding values. Example: is_features=0:3;1:1. This means that in feature set 0, features 1 + 2 are enabled and in feature set 1, feature 1 is enabled.
- user = username defined in Manage Keys / 3rd Party Extensions (optional)
- password = password defined in Manage Keys / 3rd Party Extensions (optional)

Note: Due to limitations in the maximum number of characters in SWREG's user_text field, the name of the arguments above have been abbreviated.

For example:

<https://usd.swreg.org/cgi-bin/s.cgi?s=46994&p=46994TESTPROD&v=0&d=0&q=1&t=pid%3d4%26mj%3d4%26mn%3d0%26vn%3dSWREG&x=1>

With the steps above completed, place a test order. When the order is placed, the following will occur:

-
- A new user will be automatically added to the QLM database based on the information collected during the ordering process.
-
- An activation key will be created in the QLM database and associated to the user.
-
- An email will be sent to the customer with the activation key.

UltraCart

Quick License Manager integrate with UltraCart's ordering system to generate activation codes automatically.

To have UltraCart invoke QLM during an order, do the following:

From within your UltraCart account:

From Item Management -> Items

-
- Locate your Product and open the Item Editor for this product
-
- In the Item Editor, select the **Digital Delivery** tab
-
- In the **Activation Codes** section of the Digital Delivery tab:
-
- Select **Retrieve Real-time**
-
- Enter the following URL (change the 'yourserver' name to your own)
`http://yourserver/qlm/qlmservice.aspx/GetActivationKey?is_vendor=ultracart&is_productid=1&is_majorversion=1&is_minorversion=0&is_qlmversion=5.0.00&is_features=0 ;`
- Where
 - is_vendor = ultracart
 - is_productid = your product id as defined in QLM
 - is_majorversion = your product's major version as defined in QLM
 - is_minorversion = your product's minor version as defined in QLM
 - is_qlmversion = 5.0.00
 - is_features = semicolon separated list of feature sets and their corresponding values.
Example: is_features=0:3;1:1. This means that in feature set 0, features 1 + 2 are enabled and in feature set 1, feature 1 is enabled.
- Enter a value in the **Shared Secret** field. Remember this value as it needs to be set in QLM as well.

Next configure QLM to include the UltraCart provider as follows:

From Manage Keys-> License Management tab

-
- Select **Tools -> eCommerce Providers**
-
- Click the Add button and enter the following values:
 - Name: UltraCart
 - Class: Qlmsvc.UltraCart
 - Dll: UltraCart.dll
 - User:
 - Password: ***Shared Secret*** entered in *UltraCart Activation Code* section

Edit your QLM server's Web.config file and set the following setting:

`<setting name="defaultVendor" serializeAs="String">`

<value>UltraCart</value>

</setting>

With the above steps completed, place a test order. When the order is placed, the following will occur:

-
- A new user will be automatically added to the QLM database based on the information collected during the ordering process.
-
- An activation key will be created in the QLM database and associated to the user.

Adding a new eCommerce Provider

QLM Professional supports several eCommerce providers. If QLM does not support your eCommerce provider, you can either extend QLM to support your provider or contact our Professional Services to receive a quote for supporting your eCommerce provider.

Creating your eCommerce Provider plugin

The sample program Regnow located in:

`%Public%\Documents\Quick License Manager\Samples\QLMPro\Windows\Regnow`

provides a starting point to create your own eCommerce provider plugin.

The eCommerce provider plugin serves 2 purposes: (a) to extract information sent by your eCommerce provider during the purchase process and map this information to QLM fields and (b) to format the output of the license key request sent by your eCommerce provider using the syntax expected by your eCommerce provider.

Configuring data extraction from eCommerce provider into QLM Database

Upon an item purchase, eCommerce providers can typically invoke a License Server and provide details about the order. The information provided includes data such as customer name, email address, product ordered, number of licenses, etc.. In order to capture this information and update the QLM database accordingly, the plugin needs to map the eCommerce provider data into QLM fields. A set of properties can be overwritten in the plugin class to customize this mapping. The properties that can be overwritten are:

- Order ID
- OrderStatus
- ReceiptID
- CustomerEmail
- CustomerName
- CustomerCompany
- CustomerAddress1
- CustomerAddress2
- CustomerCity
- CustomerState
- CustomerZip
- CustomerCountry
- CustomerPhone
- CustomerIP
- CustomerNotes
- Quantity
- MaintenancePlan

Configuring the format of the license key

Upon purchase of an item, eCommerce providers typically expect a license key to be returned when they invoke the QLM License Server. The format of the returned key or keys differs for each eCommerce provider.

You can customize via the QLM plugin the format of the returned key to meet your eCommerce provider's requirements.

A set of methods can be overwritten in the plugin class to customize this syntax. The methods that can be overwritten are:

- WriteStart
- WriteEnd

WriteKeyStart
WriteKeyEnd
WriteOrderInfo
WriteMaintenanceRenewalResponse
WriteUpgradeLicenseResponse
WriteError

Configuring authentication

eCommerce providers connect to the QLM License Server via an http request. To ensure that hackers do not contact your License Server easily, you can provide a username / password on the url command line. Since the License Server is invoked directly from your eCommerce provider, end users will never be able to intercept that URL and discover the password.

The default behavior of QLM supports validating a user/password by comparing the command line arguments with a user/password that you can configure and define in the QLM Console. You can also customize the authentication mechanism by overwriting the following method:

AuthenticateUser

Deployment of your plugin

Once you have completed development of your plugin, follow the procedure below to install it and register it:

- Deploy the DLL to the bin folder on your web server, in the same folder as the ***QlmCommerceProvider.dll*** file.
- Launch the QLM Console
- Click on Manage Keys / Tools / eCommerce Providers
- Click on the Add button and enter the data associated with your provider
- Fill in all the fields. For help on each field, click on the text field and read the help in the loght bulb section
- Click OK

Testing your plugin

Once the above steps are completed, you can test your plugin. Most eCommerce providers support a test mode where you can submit a dummy order. To help diagnose problems with your plugin, configure the QLM License Server to log verbose events by updating the **loggingLevel** setting to **15**

When a test is performed from your eCommerce provider, you can view details about the request from the QLM Events Viewer. To launch the Events Viewer, click on Manage Keys / Tools / View Server Event Log.

Check for Updates

QLM provides a framework that allows you to implement a "Check for Updates" feature for your software. The "Check for Updates" feature allows you to automatically inform your users when a new version of your software is available to download.

Below are the steps required to implement a Check for Updates feature in your application:

- In the Define Product screen, click on the Latest Version tab and specify the details of the latest version of your software:
 - Latest Version: Enter the version number of your latest release. Example: 3.1.0
 - URL the latest version: Enter a URL from where the user can download the latest version. This URL can be used from your application to either automatically download your application or simply display the URL to the user.
 - Notes about latest version: Enter notes that you would like to display to the user if QLM detects that a new version is available.
- In QLM, click on Manage Keys / Sites. In the License Server tab, select the appropriate profile and click on "Upload products to License Server" to upload your product data to the server.
- In your application, create a button or a timer based routine that calls the GetLatestVersion function (see API Reference for details).
- Compare the server's version with the installed version and prompt users with the option to upgrade their version if appropriate.

The QLM Check For Updates Sample shows how to implement the Check for Updates feature in your application. The sample is located in:

- %Public%\Documents\Quick License Manager\Samples\qlmpro\Windows\DotNet\Basic\C#\QlmCheckForUpdates

Terminal Server

QLM Professional can limit the number of instances of your application running on a Terminal Server. To restrict the number of instances on a Terminal Server, you must set the "Floating Seats" property when creating an Activation Key.

Below are the steps required to specify the number of allowed terminal server instances:

- Click on the Manage Keys tab.
- Click on the Create button.
- Specify the Floating Seats property to the number of instances allowed to run on the Terminal Server.
- Click OK

In your code, you must also set the `LimitTerminalServerInstances` property to *true*. The default value is *false*

Note that an instance is uniquely identified by a session id and a user id. For example, if only 1 instance is allowed, the same user can still launch your application multiple times within the same Terminal Server session. That same user will not be able to start another Terminal Server session and start another instance of your application.

Scheduled Tasks

QLM can schedule and execute tasks such as emailing notifications to your customers or displaying alerts when the QLM database is updated. QLM includes one built-in email scheduled task designed for implementing a Maintenance Plan for your product and several alert type tasks. You can use these tasks as is, customize them to suit your needs, or create other scheduled tasks that meet your special needs.

Note that in order for tasks to run, the Quick License Manager Agent must be running.

Below are the steps required to create scheduled tasks:

- Click on the Manage Keys tab.
- Click on the Scheduled Tasks button.
- Click on Add.
- If you have multiple web servers, select the License Server profile to use. Otherwise, select the Default profile.
- Select the Search to execute. The scheduled task will be executed on each record returned by the search.
- Specify how often the task will run.
- To configure an Email notification task, click on the Email tab and check the Enable Email Notification box, then fill in all the remaining fields.
- To configure an Alert notification task, click on the Alert tab and check the Enable Alert box, then fill in all the remaining fields. The Message fields supports variables such as %FullName% or %ActivationKey%.

Note that QLM uses your Outlook 2003 or 2007 client to send emails. You need to be logged in to the system for email notifications to work.

Maintenance Plan

QLM provides a complete framework for implementing a maintenance plan for your software. When a customer activates a license key, the QLM License Server validates that the license key corresponds to the version the customer is running. For example, if a customer is running version 6.0 of your software but has an Activation key for version 5.0, the QLM License Server will fail to activate the license.

However, if a customer purchases a maintenance plan for your software, the QLM License Server will allow activation of a recent version of your software with an activation key for a previous version. For example, if a customer is running version 6.0 of your software but has an Activation key for version 5.0, the QLM License Server will successfully activate the license if the maintenance plan is active and the MaintenancePlanRenewal date is greater than the Release Date of the new version of your product. The Release Date of your product can be defined on the Define Product page in the QLM Console.

The QLM database stores maintenance plan information in the MaintenanceRenewalDate field. If the maintenance plan is not enabled, this field will be empty. If the maintenance plan is enabled, the field will be set to the expiry date of the maintenance plan.

The MaintenanceRenewalDate can be set during the order process or manually via the QLM console.

To enable the maintenance plan during the ordering process, you can invoke the following URL:

```
http://yourserver/qlm/qlmservice.aspx/EnableMaintenancePlan?is_vendor=<vendor>&is_user=<user>&is_pwd=<pwd>
```

where:

- <vendor> is the name of your vendor (see list of supported eCommerce providers)
- <user> is a user to use for authentication (as defined in Manage Keys / 3rd Party Extensions)
- <pwd> is a password to use for authentication (as defined in Manage Keys / 3rd Party Extensions)

The maintenance plan date is set to 365 days after the Order Date. You can control this setting from the web.config file of the QLM License Server.

Alternatively, you can customize your eCommerce provider order form to publish a new field during the POST to specify whether the maintenance plan was purchased. The name of the field is typically different for each eCommerce provider. The table below describes the name of this field for each of the supported eCommerce providers:

Provider	Field Name
Digibuy	maintenance
Regnow	yearly_maintenance_plan
Paypal	MaintenancePlan
ShareIt	yearly_maintenance_plan
Plimus	is_maintenance_plan

To set the maintenance plan using the QLM console:

- For a new activation key, click on Manage Keys / Licenses / Create and check the Maintenance Plan check box.
- For an existing activation key, click on Manage Keys / Licenses / Edit then specify a Renewal Date for the Maintenance Plan.

Maintenance Plan Email Notification

To remind customers to renew the maintenance plan, you can schedule a task that will send your customers a reminder email prior to the expiry of the maintenance plan. For more details, see the help on Scheduled Tasks.

Place an order to renew a maintenance plan

When a maintenance plan expires, you can renew the maintenance plan from your eCommerce provider's ordering system by invoking the following URL:

`http://yourserver/qlm/qlmservice.asmx/RenewMaintenancePlan?&is_avkey=<activationKey>&is_vendor=<vendor>`

where:

- `<activationKey>` is the activation key of the customer
- `<vendor>` is the name of your vendor (see list of supported eCommerce providers)

Subscription licensing

If you sell your software as a subscription, QLM can create license keys that expire at a specific date or after a set duration. This is identical to trial keys.

To create a license key with an expiry date, you can use one of the following methods:

1. The QLM Console (Manage Keys / Create)
2. The QLM API (CreateActivationKeyWithExpiryDateEx).
3. The Http method GetActivationKeyWithExpiryDate, typically invoked directly from your ecommerce provider. See Http methods in the help for more details.

Once a subscription expires, if your customer does not renew the subscription, the license expires and your software no longer runs.

If your customer decides to renew the subscription, you can renew the license without sending your customer a new license key. The customer simply needs to reactivate their license to extend it to the new expiry date.

Extending a subscription can be performed in 3 different ways:

1. From the QLM Console, under Manage Keys / Renew Subscription
2. By using the QLM API RenewSubscription from your application.
3. By using the Http method RenewSubscriptionHttp from your ecommerce provider. See Http methods in the help for more details.

Affiliates

If you use affiliates to sell your product, QLM allows you to manage your affiliates' sales. The QLM Portal provides a web interface for your affiliates to generate license keys for their customers. You can configure the following restrictions for Affiliates:

- Maximum number of trial keys per system
- Maximum number of permanent keys per system
- Maximum total keys
- Maximum activations per key

In addition, you can control what operations an Affiliate can perform from the QLM Portal:

- Creating new keys
- Activating keys
- Releasing keys
- Deleting keys
- Creating customers
- Deleting customers
- Exporting keys
- Setting the Expiry Duration of a new key
- Settings the Expiry Date of a new key
- Settings the Maintenance Plan option
- Setting the Generic license option

Registering your license

The QLM Portal is an add-on to QLM, it is not a replacement for the QLM Management Console. It provides a subset of the features available in the QLM Management Console.

To register your license and enable the QLM Portal add-on:

- Start the QLM Application
- Go to the Manage Keys tab
- Click on Sites
- Select your License Server profile
- In the Portal License Key field, enter your QLM Portal license key and click on Register. Note that the QLM Portal License Key is not the same as your QLM Professional or Enterprise license key. If you have not purchased the QLM Portal add-on, you can do so from our web site.

Installing the QLM Portal

The QLM portal can be installed via the setup (QlmLicenseServerSetup.exe) or manually.

If you can run a setup program on your web server, run the QlmLicenseServerSetup.exe on your server and make sure that the QLM Portal feature is selected.

If you cannot run a setup program on your web server, following are the manual steps to install the QLM Portal:

- Create a virtual directory on your web server called: QlmPortal
- Upload all the files in the %Public%\Public Documents\Quick License Manager\DeployToServer folder to the QlmPortal virtual directory
- Enable ASP.NET 4.0 for the virtual directory.
- Customize the following appSettings in the web.config file as follows:
 - communicationEncryptionKey

- adminEncryptionKey
- sqlSyntax
- webServiceUrl

- Additionally, you should customize the connectionStrings settings in the web.config file.

The QLM portal uses ASP.NET forms authentication to validate users. To enable ASP.NET forms authentication, the QLM database needs to be updated to maintain authentication information. To upgrade your QLM database, run the sql2005.aspnet.sql script located in the following folder (if you installed the QLM Portal via the QlmLicenseServerSetup.exe, this step can be skipped):

%Public%\Public Documents\Quick License Manager\DeployToServer\QlmLicenseServer\Db

Accessing the QLM Portal

The first step in setting up a portal is to create the Administrator's user account. To do so, start the QLM Application and:

- Go to the Manage Keys tab.
- Click on User Accounts in the Portal group.
- Click on the Add button to create a new user account.
- Fill in all the fields as required. When asked to associate a User Profile with the user account, if you select None, the user will be able to create an unlimited amount of keys. If you select a User Profile, the user will be able to create as many keys as allowed for this specific User Profile.

Now that you have an account, you may login to the portal at the following URL:

<http://yourserver/qlmportal/qlmportal.aspx>

Once logged in, you can create license keys for your products as well as add new customers.

To create a user account for your affiliate:

- Go to the Manage Keys tab.
- Click on User Profiles in the Portal group.
- Click on the Add button to create a new user affiliate. You can control how many license keys an affiliate can create and for which products they can create license keys.
- Once the affiliate created, click on User Accounts in the Portal group.
- Click on the Add button to create a new user account for your affiliate and select the corresponding Affiliate from the Affiliate dropdown.
- Provide your affiliate with the URL to the portal and his account information.

On the QLM Portal, affiliates can only see customers that they are associated to. If a customer is created from the QLM Portal, the customer is automatically associated to the logged in affiliate. If a customer is created from the QLM application Manage Customers tab, you can associate the customer to the affiliate when the customer is created or at any other time by clicking on the Manage Customers / Edit option.

Server Event Log

When errors occur in the QLM License Server, QLM logs errors messages to the QLM database. These error messages can be viewed from the QLM Console by clicking on Mange Keys / Tools / View Server Event Log. QLM has several levels of logging. QLM can log errors, warnings and information. You can control the logging level by editing the loggingLevel setting in web.config file of the QLM License Server. The possible values for the logging level are:

- Error=1
- Warning=2
- Information=4
- Verbose=8

These settings can be combined together as follows:

To log errors and warnings, set the value to: 3

To log errors, warnings and information, set the value to: 7

To log errors, warnings, information and verbose messages, set the value to: 15

Illegal Computers

QLM can track illegal computers that connect to the License Server and logs information about these computers in the database. An illegal computer is defined as a computer that has a valid license key but whose license key is (a) not in the database or (b) in the database but registered for another computer. This situation can occur if a user with a valid license key requests that his license be released from computer A claiming to have uninstalled your program from computer A. If the user subsequently attempts to connect to the License Server via computer A, QLM detects this computer as an illegal computer and logs it in the database. There are two ways to enable illegal computers detection in your application:

- The QlmCustomerSite provides a web page that your application can connect to by invoking a url. For example:
- `http://yourserver/QlmCustomerSite/qlmlicenseinfo.aspx/?is_avkey=AGGI0U0Q00NSSUY8EH31U1TZ4&is_cpkey=UAJD0M0600PBIUU28NKH1A12JM&is_cpid=MYPC&is_cpname=MYPC&is_qlmversion=5.0.00`
- where:
- `is_avkey` specifies the user's activation key
- `is_cpkey` specifies the user's computer bound key
- `is_cpid` specifies the unique computer identifier
- `is_cpname` specifies the computer name
- `is_qlmversion` specifies the version of the QLM engine
- When your application is launched, connect to the page above to display license information to your end user. In addition to providing licensing information to the user, this page will detect illegal computers and log them in the database. The QlmCustomerSite can be found in the %Public%\Public Documents\Quick License Manager\DeployToServer folder. The deployment procedure for this portal is identical to the one for the QLM License Server. If you have installed the QLM License Server using the setup program QlmLicenseServerSetup.exe, then the QlmCustomerSite is already deployed on your server.
- The QLM API (in QlmLicenseLib.dll) includes a new method called *IsIllegalComputer* to detect illegal computers. This method should be ideally be called every time your application is launched.

To view illegal computers, click on the Manage Keys tab then select the Illegal Computers button. Note that QLM does not prevent users from running your application if an illegal computer is detected.

Deactivating a license

Deactivating your customer's license from one computer so that it can be activated on another can be done in one of two ways: (a) via the QLM Application or (b) via the QLM API.

To release a license via the QLM Application:

1. In the QLM Application, under Manage Keys, locate the license to transfer and select it
2. Click on the **Release** button

It is recommended to use the `QlmLicense.IsIllegalComputer` method in your application to detect illegal usage of a license.

To release a license via the QLM API:

1. Call the `QlmLicense.ReleaseLicense` method
2. Call the `QlmLicense.DeleteKeys` method

Releasing a QLM license via the QLM API is the preferred option as it allows you to delete license keys stored on the end user system as well as releasing the license in the QLM database.

Additionally, it is recommended to integrate releasing a license in your uninstaller.

Calling the QLM .NET API from VB or C++

The QLM API that connects to the QLM License Server was developed with .NET. If your application was developed in C++, VB6 or any other unmanaged code that cannot call .NET API directly, you may still use the QLM API via the COM interface.

To enable the QLM API to be called as a COM interface, you need to do the following:

- **Generate a type library to be referenced by your code**
- `regasm /tlb "<fullpath>\QlmLicenseLib.dll"`
- **Register the QlmLicenseLib.dll as a COM object**
- `regasm /codebase "<fullpath>\QlmLicenseLib.dll"`

QLM Customer Site

The QLM Customer site is a collection of web pages that you can integrate to your site to automate some processes and provide self-help to your customers. The available pages are:

- `QlmLicenseInfo.aspx` - Displays license information to your end users
- `QlmRegistrationForm.aspx` - A registration form that collects user information and generates a trial key for a given product
- `QlmReleaseLicense.aspx` - A form that your users can fill out to de-activate a license from a given computer
- `QlmWebActivation.aspx` - A form that your users can fill out to activate a license from a given computer. This is particular useful if the computer that needs to activate a license is not connected to the internet.

The QLM Customer Site is deployed alongside the QLM License Server in the `QlmCustomerSite` folder. If you deploy the QLM License Server manually, you must deploy the `QlmCustomerSite` (in the `DeployToServer` folder) and configure the `QlmCustomerSite` as an IIS Application.

PAGE	ARGUMENT	DESCRIPTION
QlmLicenseInfo.aspx		
	<code>is_avkey</code>	Activation Key of the end user
	<code>is_pckey</code>	Computer Key of the end user.
	<code>is_pcid</code>	Computer ID of the end user.
	<code>is_pcname</code>	Computer Name of the end user.
	Example:	<code>http://quicklicensemanager.com/QlmCustomerSite/qlmlicenseinfo.aspx?is_avkey=AXK80-60R00-GHJ3S-I862Y-1I1UR-AQDV2&ispckey=UTME0D0P00BZ74ZW8SNV1N1S5D&is_pcid=123</code> ;
QlmRegistrationForm.aspx		
	<code>is_productname</code>	Name of the product.
	<code>is_productid</code>	ID of the product.
	<code>is_majorversion</code>	Major version of the product.
	<code>is_minorversion</code>	Minor version of the product.
	<code>is_expduration</code>	Duration of the license, i.e. number of days after which the license will expire
	<code>is_expdate</code>	Date at which the license expires. Default date format is: yyyy-MM-dd. The format can be changed from the web.config file.
	<code>is_emailfrom</code>	When sending email, specify the email

	address of the sender.
is_emailsubject	Customize the subject of the email. Variables are allowed. See supported variables below.
is_confirmationmessage	The message that is displayed to the user upon successful registration. Variables are allowed.
Variables	The following variables can be used in the appropriate fields: %ActivationKey% %FullName% %Email% %EmailFrom% %ProductName% %MajorVersion% %MinorVersion%
Example	http://quicklicensemanager.com/QlmCustomerSite/qlmregistrationform.aspx?is_productname=MyProduct&is_productid=1&is_majorversion=1&is_minorversion=0&is_expduration=10&is_emailfrom=support@soraco.co&is_emailsubject=Your Download of %ProductName%&is_confirmationmessage=Your License information was emailed to: %Email% ;

QlmReleaseLicense.aspx

is_avkey	Activation Key of the end user. This argument is optional.
is_pcid	Computer ID of the end user. This argument is optional.
Example	http://quicklicensemanager.com/QlmCustomerSite/QlmReleaseLicense.aspx?is_avkey=AXK80-60R00-GHJ3S-I862Y-1I1UR-AQDV2&is_pcid=123;

QlmWebActivation.aspx

is_avkey	Activation Key of the end user. This
----------	--------------------------------------

argument is optional.

is_pcid

Computer ID of the end user. This argument is optional.

Set is_file to 1 to have this page generate a digitally signed license file (for QLM Enterprise users building cross platform apps). When generating a license file, you can customize the following settings in the web.config file of QlmCustomerSite:

is_file

- offlineActivationSuccessMessage: the message that is displayed upon successful generation of the license file.
- licenseFileName: the name of the generated license file.
- licenseFileName: the name of the generated license file.

Example for generating a computer key:

http://quicklicensemanager.com/QlmCustomerSite/QlmWebActivation.aspx?is_avkey=AXK80-60R00-GHJ3S-I862Y-1I1UR-AQDV2&is_pcid=123;

Example for generating a digitally signed license file:

http://quicklicensemanager.com/QlmCustomerSite/QlmWebActivation.aspx?is_avkey=AXK80-60R00-GHJ3S-I862Y-1I1UR-AQDV2&is_pcid=123&is_file=1;

Distribute your application

The sections below describe the options to include the QLM required DLLs in your application. Select the most appropriate option based on the type of setup that you use to deploy your application.

Using the QLM .NET API from VB6 or unmanaged C++

If you have included a reference to QlmLicenseLib.dll in your VB6, unmanaged C++ or any other non .NET based application, your setup must perform the following operations:

32 bit applications

- **Generate a type library to be referenced by your code**
 - `%windir%\Microsoft.NET\Framework\v2.0.50727\regasm.exe /tlb "<fullpath>\QlmLicenseLib.dll"`
- **Register the QlmLicenseLib.dll as a COM object**
 - `%windir%\Microsoft.NET\Framework\v2.0.50727\regasm.exe /codebase "<fullpath>\QlmLicenseLib.dll"`

64 bit applications

- **Generate a type library to be referenced by your code**
 - `"%windir%\Microsoft.NET\Framework64\v2.0.50727\regasm.exe" /tlb "<fullpath>\QlmLicenseLib.dll"`
- **Register the QlmLicenseLib.dll as a COM object**
 - `"%windir%\Microsoft.NET\Framework64\v2.0.50727\regasm.exe" /codebase "<fullpath>\QlmLicenseLib.dll"`

Distribute your application using a manual procedure

Manual steps to install Quick License Manager files

If you are not using Windows Installer to distribute your application, the following files must be included in your application:

IsLicense50.dll: not shared, to be installed in the same folder as your application

QlmLicenseLib.dll: not shared, to be installed in the same folder as your application

QlmControls.dll: not shared, to be installed in the same folder as your application

If you are targeting x64 bit systems, you will need to conditionally install the proper IsLicense50.dll depending on the target platform.

The x64 bit version of IsLicense50.dll is located in the Redistrib\x64 folder.

Application Type	Files to include
Windows Forms .NET 2.0 or higher	redistrib\x86\IsLicense50.dll redistrib\x64\IsLicense50.dll redistrib\.net2.0\QlmLicenseLib.dll redistrib\.net2.0\QlmControls.dll
ASP.NET 2.0 or higher Excel 2003 or higher MS-Access 2003 or higher Outlook Add-in VB6	redistrib\.net2.0\QlmLicenseLibEmb\QlmLicenseLib.dll redistrib\.net2.0\QlmControls.dll
C++	redistrib\x86\IsLicense50.dll redistrib\x64\IsLicense50.dll redistrib\.net2.0\QlmLicenseLib.dll redistrib\.net2.0\QlmControls.dll

Localization

The QLM .NET API calls may return messages relating the status of a license key validation. By default, all messages returned by the QLM .NET API are in English. The QLM .NET API also supports many languages such as Spanish, Italian and German messages. To configure your application to display the proper message depending on the language of your customer's system, you need to:

- Locate the localization folders in the Quick License Manager "redistrib\Localization" folder.
- When you deploy your application, copy these folders to the same location as the QlmLicenseLib.dll
- For example, if your customer's system is running Spanish OS, copying the "es" folder will result in the Spanish resources to be automatically used.

Alternatively, if you would like to force a specific language, you need to add the following call to your application prior to initializing any UI:

```
System.Threading.Thread.CurrentThread.CurrentUICulture = new  
System.Globalization.CultureInfo("es-ES");
```

Additionally, the QLM .NET Controls as well as the QLM License Wizard are localized. The QlmControl.resources.dll contains the localized resources for the QlmControls.dll whereas QlmLicenseWizard.resources.dll contains the localized resources for the QLM License Wizard.

Troubleshooting License Server

If you have errors connecting to the License Server, check the following troubleshooting tips:

- Try to connect to the License Server from the browser by typing the following URL in the browser: <http://server/qlm/qlmservice.asmx>
- Determine the credentials of the user running the virtual directory. If you have installed the QLM License Server using the QlmLicenseServerSetup.exe, an application pool was created called Quick License Manager. The application pool runs by default with the credentials of the NETWORK SERVICE account. The NETWORK SERVICE account will therefore need Modify permissions on the following folders:
 - C:\WINDOWS\Microsoft.NET\Framework\v2.0.50727\Temporary ASP.NET Files
 - C:\Windows\Temp
 - C:\Program Files\Soraco\QuickLicenseMgr\QlmWebSvc\Db
- If you have installed the QLM License Server manually and have not created an application pool, make sure that the anonymous user (IUSR_XXX and IWAM_XXX) have Modify permissions on the following folders:
 - C:\WINDOWS\Microsoft.NET\Framework\v2.0.50727\Temporary ASP.NET Files
 - C:\Windows\Temp
 - C:\Program Files\Soraco\QuickLicenseMgr\QlmWebSvc\Db (or wherever you have installed the database file).
- If you are using MS-Access as a database, verify that the qlm.accdb file on the web server is not read-only. Additionally, verify that the user configured in the IIS QLM virtual directory has Write privileges on the folder where the qlm.accdb file is located. If you are using an Application Pool (recommended), verify that the user who is defined in the Identity tab of the Application Pool has Write privileges on the folder where qlm.accdb is located.
- Verify that the communicationEncryptionKey and adminEncryptionKey between the client and the License Server match. In the License Server, edit the web.config and note the value of communicationEncryptionKey and adminEncryptionKey. In the QLM Console, click on Manage Keys, select License Server / Sites then on the Encryption Keys tab, verify that both values match.

QLM API Reference

The QLM API is organized in 4 different categories:

- The **Server Application API** includes all methods that you invoke from your application and that communicate with the QLM License Server.
- The **License Server Management API** includes all methods that are administrative type of functions that you typically invoke from your server or from a third-party process in order to integrate QLM with 3rd party tools.
- The **Client Side API** includes all methods that are used to perform operations on license keys locally on the end user system without contacting the License Server.
- The **License Server HTTP methods (REST)** includes all methods that can be invoked by simply performing an http request (without using our API). These methods are typically invoked from your eCommerce provider during the purchasing process. Note that all other methods mentioned above cannot be called directly via SOAP. In order to communicate with the QLM License Server, you need to use the QLM API methods that are exposed via the QLMLicenseLib.dll.

License Server Application API

The functions listed in this section are functions that you need to use within your application in order to interface with the QLM License Server.

Before calling any of the functions below, you need to set the `communicationEncryptionKey` property of the `QlmLicense` object to the value specified in your Site Properties (Manage Keys tab / Site). If you are developing an application in .NET, it is highly recommended that you obfuscate your code, and more specifically that you encrypt sensitive strings such as the `communicationEncryptionKey`.

The QLM License Server provides a more extended set of APIs that you may want to use for managing your license keys or for integrating licensing with any other internal process that you may have. The extended set of functions is described under the **License Server Management API** section.

ActivateLicense

Activates a license key over the internet. To use a proxy server, you must set the UseProxyServer, ProxyUser, ProxyDomain and ProxyPassword properties prior to calling this function.

C#: void ActivateLicense (string webServiceUrl, string activationKey, string computerID, string computerName, string qlmVersion, string userData1, out string response)

Parameters

webServiceUrl - URL to the QLM License Server.

activationKey - the license key to activate

computerID - the unique computer identifier

computerName - the name of the computer, not required but recommended.

qlmVersion - the version of the QLM Engine

userData1 - User data to associate with the license key

response - XML fragment containing the result of the call. The Xml fragment schema is as follows:

- <?xml version='1.0' encoding='UTF-8'?>
- <QuickLicenseManager>
- <pckey>C06C4C90A497F091C2F080501000C076A0578E</pckey>
- <userCompany>My Company</userCompany>
- <userFullName>John Smith</userFullName>
- <userEmail>john@smith.com</userEmail>
- </QuickLicenseManager>

In the event of an error, the XML fragments returns:

- <?xml version='1.0' encoding='UTF-8'?>
- <QuickLicenseManager>
- <error>Details about the error</error>
- </QuickLicenseManager>

ActivateLicenseByField

Activates a license key over the internet, binds it to a specific user and returns a computer bound license key. To use a proxy server, you must set the UseProxyServer, ProxyUser, ProxyDomain and ProxyPassword properties prior to calling this function.

This function can be used to activate a license when the Activation Key is not known. For example, you can use this function to activate a license based on an Order ID. To do so, you set the fieldName argument to "OrderID" and set the fieldValue argument to the value of the Order ID.

By default, the supported fields are: OrderID and ReceiptID.

You can modify the list of supported fields by updating the activationByFieldAllowedFields setting in the QLM License Server's web.config file.

C#: ActivateLicenseByField(string webServiceUrl, string fieldName, string fieldValue, string email, string computerID, string computerName, string qlmVersion, string userData1, string affiliateID, out string response)

Parameters

webServiceUrl - URL to the QLM License Server

fieldName - Name of the field used to locate the Activation Key to activate

fieldName - fieldValue - Value of the field used to locate the Activation Key to activate

email - Email address of user that owns the license

computerID - Unique computer identifier

computerName - Name of computer

qlmVersion - Version of the QLM Engine to use

userData1 - User data to associate with the license key

affiliateID - ID of affiliate

response - XML fragment containing the result of the call. The Xml fragment schema is as follows:

- <?xml version='1.0' encoding='UTF-8'?>
- <QuickLicenseManager>
- <pckey>C06C4C90A497F091C2F080501000C076A0578E</pckey>
- <userCompany>My Company</userCompany>
- <userFullName>John Smith</userFullName>
- <userEmail>john@smith.com</userEmail>
- </QuickLicenseManager>

In the event of an error, the XML fragments returns:

- <?xml version='1.0' encoding='UTF-8'?>
- <QuickLicenseManager>
- <error>Details about the error</error>
- </QuickLicenseManager>

ActivateLicenseEx

Activates a license key over the internet. To use a proxy server, you must set the UseProxyServer, ProxyUser, ProxyDomain and ProxyPassword properties prior to calling this function.

C#: void ActivateLicenseEx (string webServiceUrl, string activationKey, string computerID, string computerName, string qlmVersion, string userData1, string affiliateID, out string response)

Parameters

webServiceUrl - URL to the QLM License Server.

activationKey - the license key to activate

computerID - the unique computer identifier

computerName - the name of the computer, not required but recommended.

qlmVersion - the version of the QLM Engine

userData1 - User data to associate with the license key

affiliateID - ID of affiliate

response - XML fragment containing the result of the call. The Xml fragment schema is as follows:

- <?xml version='1.0' encoding='UTF-8'?>
- <QuickLicenseManager>
- <pckey>C06C4C90A497F091C2F080501000C076A0578E</pckey>
- <userCompany>My Company</userCompany>
- <userFullName>John Smith</userFullName>
- <userEmail>john@smith.com</userEmail>
- </QuickLicenseManager>

In the event of an error, the XML fragments returns:

- <?xml version='1.0' encoding='UTF-8'?>
- <QuickLicenseManager>
- <error>Details about the error</error>
- </QuickLicenseManager>

ActivateLicenseDialog

Creates a form for the user to enter his contact information and the activation key. When submitted, the user is added to the database and his license is activated.

C#: public bool ActivateLicenseDialog(string webServiceUrl, string computerID, string userData1, Control parentWindow, out string response)

Parameters

webServiceUrl - URL to the QLM License Server.

computerID - Unique identifier for the computer

userData1 - user data to associate to this license

parentWindow - parent of the activation form window

response - XML fragment containing the result of the call. The Xml fragment schema is as follows:

- <?xml version='1.0' encoding='UTF-8'?>
- <QuickLicenseManager>
- <pckey>C06C4C90A497F091C2F080501000C076A0578E</pckey>
- <userCompany>My Company</userCompany>
- <userFullName>John Smith</userFullName>
- <userEmail>john@smith.com</userEmail>
- </QuickLicenseManager>

In the event of an error, the XML fragments returns:

- <?xml version='1.0' encoding='UTF-8'?>
- <QuickLicenseManager>
- <error>Details about the error</error>
- </QuickLicenseManager>

ActivateLicenseForUser

Activates a license key over the internet, binds it to a specific user and returns a computer bound license key. To use a proxy server, you must set the UseProxyServer, ProxyUser, ProxyDomain and ProxyPassword properties prior to calling this function.

C#: ActivateLicenseForUser(string webServiceUrl, string activationKey, string email, string computerID, string computerName, string qlmVersion, string userData1, string affiliateID, out string response)

Parameters

webServiceUrl - URL to the QLM License Server

activationKey - Activation key

email - Email address of user that owns the license

computerID - Unique computer identifier

computerName - Name of computer

qlmVersion - Version of the QLM Engine to use

userData1 - User data to associate with the license key

affiliateID - ID of affiliate

response - XML fragment containing the result of the call. The Xml fragment schema is as follows:

- <?xml version='1.0' encoding='UTF-8'?>
- <QuickLicenseManager>
- <pckey>C06C4C90A497F091C2F080501000C076A0578E</pckey>
- <userCompany>My Company</userCompany>
- <userFullName>John Smith</userFullName>
- <userEmail>john@smith.com</userEmail>
- </QuickLicenseManager>

In the event of an error, the XML fragments returns:

- <?xml version='1.0' encoding='UTF-8'?>
- <QuickLicenseManager>
- <error>Details about the error</error>
- </QuickLicenseManager>

AddUser

Adds a new user. To use a proxy server, you must set the UseProxyServer, ProxyUser, ProxyDomain and ProxyPassword properties prior to calling this function.

C#: void AddUser(string webServiceUrl, string customerName, string customerEmail, string customerPhone, string customerFax, string customerMobile, string customerCompany, string customerAddress1, string customerAddress2, string customerCity, string customerState, string customerZip, string customerCountry, string customerIP, string customerNotes, bool includeInMailList, out string response)

Parameters

webServiceUrl - URL to the QLM License Server

customerName - Full Name

customerEmail - Email address

customerPhone - Phone number

customerFax - Fax number

customerMobile - Mobile phone number

customerCompany - Company name

customerAddress1 - Address 1

customerAddress2 - Address 2

customerCity - City

customerState - State

customerZip - Zip Code

customerCountry - Country

customerIP - IP Address

customerNotes - Notes

includeInMailList - Include in mail list

webServiceUrl - URL to the QLM License Server.

activationKey - the license key to activate

computerID - the unique computer identifier

response - XML fragment containing the result of the call. The Xml fragment schema is as follows:

- <?xml version='1.0' encoding='UTF-8'?>
- <QuickLicenseManager>
- <result>Customer ABC was added successfully."</result>
- </QuickLicenseManager>

In the event of an error, the XML fragments returns:

- <?xml version='1.0' encoding='UTF-8'?>
- <QuickLicenseManager>
- <error>Details about the error</error>
- </QuickLicenseManager>

AddUserEx2

Adds a new user.

C#: bool AddUserEx2(string webServiceUrl, IQlmCustomerInfo customerInfo, bool updateIfExists, out string response);

Parameters

webServiceUrl - URL to the QLM License Server

customerInfo - customer to add

updateIfExists - update the customer information if the customer already exists in the database

response - XML fragment containing the result of the call. The Xml fragment schema is as follows:

- <?xml version='1.0' encoding='UTF-8'?>
- <QuickLicenseManager>
- <result>Customer ABC was added successfully."</result>
- <userID>99</userID>
- </QuickLicenseManager>

In the event of an error, the XML fragments returns:

- <?xml version='1.0' encoding='UTF-8'?>
- <QuickLicenseManager>
- <error>Details about the error</error>
- </QuickLicenseManager>

CreateComputerBoundTrialKey

Creates a trial activation key, then automatically activates it on the server side and returns the computer bound license key. This function is useful if you want to create trial keys from within your application. The trial period is controlled by the **trialDuration** Server Property. Server Properties can be set from the **Manage Keys / Sites / Server Properties**

page.

Prior to calling this function, you must call **DefineProduct** and set the **CommunicationEncryptionKey** property.

If you want to prevent calls to this function, set the **enableCreateComputerBoundTrial** Server Property to false. The Server Properties can be set from the **Manage Keys / Sites / Server Properties**

page.

C#: string CreateComputerBoundTrialKey(string webServiceUrl, string computerID, string computerName, string email, string features, string affiliateID, string userData1, out string response)

Parameters

webServiceUrl - URL to the QLM License Server.

computerID- Unique identifier of the computer being activated.

computerName - Friendly name of the computer being activated.

email - email address to associate to the license key - may be empty

features - Semi comma separated list of feature sets and their corresponding values.

Example: 0:1;1:2;2:3;3:6 - enables feature 1 in feature set 0, feature 2 in feature set 1, feature 1+2 (3) in feature set 4 and features 1+2+3 (6) in feature set 3.

affiliateID - ID of affiliate

userData1 - user data to associate to this license

response - XML fragment containing the result of the call. The Xml fragment schema is as follows:

- <?xml version='1.0' encoding='UTF-8'?>
- <QuickLicenseManager>
- <pckey>C06C4C90A497F091C2F080501000C076A0578E</pckey>
- <userCompany>My Company</userCompany>
- <userFullName>John Smith</userFullName>
- <userEmail>john@smith.com</userEmail>
- </QuickLicenseManager>

Return value: The returned value is the computer bound license key (ComputerKey).

GetCustomerInfo

Retrieves information about the customer associated to the specified activation key.

C#: `IQlmCustomerInfo GetCustomerInfo (string webServiceUrl, string activationKey)`

Parameters

webServiceUrl - URL to the QLM License Server.

activationKey - activation key of the customer

C# Example

- `IQlmCustomerInfo customer = license.GetCustomerInfo (webServiceUrl, "A2GM0-50K00-PYU3F-784HH-1U1V5T");`

GetDashboardInfo

Connects to the License Server and gets dashboard type information related to a site. You need to set the `QlmLicense.AdminEncryptionKey` property to call this function. To use a proxy server, you must set the `UseProxyServer`, `ProxyUser`, `ProxyDomain` and `ProxyPassword` properties prior to calling this function.

To retrieve a specific license key related information, call `GetDashboardLicenseInfo`.

C#: `IDashboardInfo GetDashboardInfo (string webServiceUrl, int recentOrders, int upcomingRenewals)`

Parameters

`webServiceUrl` - URL to the QLM License Server.

`dashboardInfo` - Dashboard info class that receives all the relevant information from the server

`recentOrders` - Define the number of days prior to today that the recent orders query returns

`upcomingRenewals` - Define the number of days after today that the upcoming renewals query returns

Return

After calling `GetDashboardInfo`, the following `IDashboardInfo` properties will be set: `TotalOrders`, `TotalCustomers`, `TodayOrders`, `YesterdayOrders`, `RecentOrders`, `UpcomingRenewals`

Example:

```
QlmLicense license = new QlmLicense ();
license.DefineProduct(1, "Demo", 1, 0, "DemoKey",
"{24EAA3C1-3DD7-40E0-AEA3-D20AA17A6005}");
DashboardInfo di = license.GetDashboardInfo
("https://qlm3.net/qlmdemov16/QlmLicenseServer/qlmservice.asmx", 10, 30);
if (ret == true)
{
    MessageBox.Show ("Total Licenses: " + di.TotalOrders);
}
```


GetDashboardLicenseInfo

Connects to the License Server and gets dashboard type information related to the specified license key. To use a proxy server, you must set the UseProxyServer, ProxyUser, ProxyDomain and ProxyPassword properties prior to calling this function.

C#: IDashboardLicenseInfo GetDashboardLicenseInfo (string webServiceUrl, string activationKey, string computerKey, string computerID)

Parameters

webServiceUrl - URL to the QLM License Server.
activationKey - Activation key
computerKey - Computer bound license key
computerID - Unique computer identifier

Return

Returns an IDashboardLicenseInfo instance.

Example:

```
QlmLicense license = new QlmLicense ();  
license.DefineProduct(1, "Demo", 1, 0, "DemoKey",  
"{24EAA3C1-3DD7-40E0-AEA3-D20AA17A6005}");  
IDashboardLicenseInfo di = license.GetDashboardLicenseInfo  
("https://qlm3.net/qlmdemov16/QlmLicenseServer/qlmservice.asmx", "AXXX", "UXXX",  
Environment.MachineName);  
if (ret == true)  
{  
    MessageBox.Show ("Total Licenses: " + di.TotalLicenses);  
}
```

GetLatestVersion

Gets the latest version of the specified product.

C#: string GetLatestVersion (string webServiceUrl, int productID, int majorVersion, int minorVersion, out string downloadUrl, out string notes, out string response)

Parameters

webServiceUrl - URL to the QLM License Server.

productID - ID of the product.

majorVersion- Major version of the product.

minorVersion - Minor version of the product.

downloadUrl - returns the url to download the latest version.

notes - returns notes about the latest version.

response - XML fragment containing the result of the call. The Xml fragment schema is as follows:

Returns the value of the latest version.

- <?xml version='1.0' encoding='UTF-8'?>
- <QuickLicenseManager>
- <result>The latest version is xyz.</result>
- <latestVersion>xyz</latestVersion>
- </QuickLicenseManager>

In the event of an error, the XML fragments returns:

- <?xml version='1.0' encoding='UTF-8'?>
- <QuickLicenseManager>
- <error>Details about the error</error>
- </QuickLicenseManager>

GetLicenseInfo

Retrieves information about an Activation Key.

C#: `ILicenseInfo GetLicenseInfo (string webServiceUrl, string activationKey, bool historyTable, out string dataSet, out string response)`

Parameters

`webServiceUrl` - URL to the QLM License Server.

`activationKey` - activation key of the customer

`historyTable` - search for the activation key in the history table

`dataSet` - data set returned by the server.

`response` - result of the call to the server.

C# Example

- `ILicenseInfo li = license.GetLicenseInfo (webServiceUrl, "A2GM0-50K00-PYU3F-784HH-1U1V5T", false, out dataSet out response);`

GetOrder

Connects to the License Server and gets the status of an order. To use a proxy server, you must set the UseProxyServer, ProxyUser, ProxyDomain and ProxyPassword properties prior to calling this function.

C#: DataRowCollection GetOrder (string webServiceUrl, string orderID, out string response)

Parameters

webServiceUrl - URL to the QLM License Server.

orderID - ID of the order

response - XML fragment containing the result of the call. The Xml fragment schema is as follows:

- <?xml version='1.0' encoding='UTF-8'?>
- <QuickLicenseManager>
- <result>Order status updated successfully.</result>
- </QuickLicenseManager>

In the event of an error, the XML fragments returns:

- <?xml version='1.0' encoding='UTF-8'?>
- <QuickLicenseManager>
- <error>The order id is not valid</error>
- </QuickLicenseManager>

Return

DataRowCollection containing all the data associated with this order.

Example:

```
QlmLicense license = new QlmLicense ();
license.DefineProduct(1, "Demo", 1, 0, "DemoKey",
"{24EAA3C1-3DD7-40E0-AEA3-D20AA17A6005}");
string response = string.Empty;
DataRowCollection drc = license.GetOrder
("https://qlm3.net/qlmdemov16/QlmLicenseServer/qlmservice.asmx", 1234, out response);
if (drc != null)
{
    foreach (DataRow dr in drc)
    {
        string activationKey = dr["ActivationKey"];
    }
}
```

GetProductInfo

Gets information about a product from the server.

C#: `IProductInfo GetProductInfo(string webServiceUrl, int productID, int majorVersion, int minorVersion)`

Parameters

`webServiceUrl` - URL to the QLM License Server.

`productID` - ID of the product.

`majorVersion` - Major version of the product.

`minorVersion` - Minor version of the product.

Returns an `IProductInfo` interface with the following properties:

- `ReleaseDate`
- `ProductName`
- `ProductID`
- `MajorVersion`
- `MinorVersion`
- `Features`
- `LatestVersion`
- `LatestVersionUrl`
- `LatestVersionNotes`

GetSubscriptionExpiryDate

Connects to the License Server and gets the subscription expiry date associated to the license. To use a proxy server, you must set the UseProxyServer, ProxyUser, ProxyDomain and ProxyPassword properties prior to calling this function.

C#: DateTime GetSubscriptionExpiryDate(string webServiceUrl, string activationKey, string computerKey, out string response)

Parameters

webServiceUrl - URL to the QLM License Server.

activationKey - The activation key to retrieve the subscription expiry date for. If this argument is set, you do not need to set the computerKey argument.

computerKey - The computer key to retrieve the subscription expiry date for. If this argument is set, you do not need to set the activationKey argument.

response - The xml fragment returned by the License Server.

GetUserData

Gets the UserData1 field for a specific license key.

C#: string GetUserData (string webServiceUrl, string activationKey, out string response)

Parameters

webServiceUrl - URL to the QLM License Server.

activationKey - activation key to query.

response - XML fragment containing the result of the call. The Xml fragment schema is as follows:

Returns the value of the userData1 field.

- <?xml version='1.0' encoding='UTF-8'?>
- <QuickLicenseManager>
- <result>Successfully executed query using filter...</result>
- </QuickLicenseManager>

In the event of an error, the XML fragments returns:

- <?xml version='1.0' encoding='UTF-8'?>
- <QuickLicenseManager>
- <error>Details about the error</error>
- </QuickLicenseManager>

GetMaintenancePlanRenewalDate

Gets the maintenance plan renewal date.

C#: `DateTime GetMaintenancePlanRenewalDate (string webServiceUrl, string activationKey)`

Parameters

webServiceUrl - URL to the QLM License Server.

activationKey - the activation key of the record.

Note that the date returned is a UTC date. In the event the maintenance plan date is not set, the return value is set to `DateTime.MinValue` which is Jan 1 0001.

GetMaintenancePlanRenewalDateByComputerKey

Gets the maintenance plan renewal date given a computer key.

C#: `DateTime GetMaintenancePlanRenewalDateByComputerKey (string webServiceUrl, string computerKey)`

Parameters

webServiceUrl - URL to the QLM License Server.

computerKey - the computer key of the record.

Note that the date returned is a UTC date. In the event the maintenance plan date is not set, the return value is set to `DateTime.MinValue` which is Jan 1 0001.

GetRemainingActivations

Connects to the License Server and determines how many activations are still available for the given activation key. To use a proxy server, you must set the UseProxyServer, ProxyUser, ProxyDomain and ProxyPassword properties prior to calling this function.

C#: int GetRemainingActivations(string webServiceUrl, string activationKey)

Parameters

webServiceUrl - URL to the QLM License Server.

activationKey - the license key to verify

GetUserDataFromActivationLog

Gets the UserData1 field for a specific license key from the ActivationLog table. The ActivationLog table is used when a license key is of type MultipleActivationsKey and more than 1 seat is associated to the license key.

C#: string GetUserDataFromActivationLog (string webServiceUrl, string activationKey, string computerID, out string response)

Parameters

webServiceUrl - URL to the QLM License Server.

activationKey - activation key to query.

computerID - Unique identifier of the computer on which the license was activated.

response - XML fragment containing the result of the call. The Xml fragment schema is as follows:

Returns the value of the userData1 field.

- <?xml version='1.0' encoding='UTF-8'?>
- <QuickLicenseManager>
- <result>Successfully executed query using filter...</result>
- </QuickLicenseManager>

In the event of an error, the XML fragments returns:

- <?xml version='1.0' encoding='UTF-8'?>
- <QuickLicenseManager>
- <error>Details about the error</error>
- </QuickLicenseManager>

IsIllegalComputer

Connects to the License Server and checks if the current computer is properly registered in the QLM database. To use a proxy server, you must set the UseProxyServer, ProxyUser, ProxyDomain and ProxyPassword properties prior to calling this function.

C#: void IsIllegalComputer (string webServiceUrl, string activationKey, string computerKey, string computerID, string computerName, string qlmVersion, out string response)

Parameters

webServiceUrl - URL to the QLM License Server.

activationKey - the activation key

computerKey - the computer bound key

computerID - the computer identifier

computerName - the computer name

qlmVersion - the version of the QLM engine

response - XML fragment containing the result of the call. The Xml fragment schema is as follows:

- <?xml version='1.0' encoding='UTF-8'?>
- <QuickLicenseManager>
- <result>The activation key is valid.</result>
- </QuickLicenseManager>

In the event of an error, the XML fragments returns:

- <?xml version='1.0' encoding='UTF-8'?>
- <QuickLicenseManager>
- <error>The activation key is not valid</error>
- </QuickLicenseManager>

IsLicenseKeyActivated

Connects to the License Server and checks if the provided license key has been activated on the specified system. To use a proxy server, you must set the UseProxyServer, ProxyUser, ProxyDomain and ProxyPassword properties prior to calling this function.

C#: bool IsLicenseKeyActivated (string webServiceUrl, string activationKey, string computerID)

Parameters

webServiceUrl - URL to the QLM License Server.

activationKey - the license key to verify

computerID - the computerID to verify

IsLicenseKeyRevoked

Connects to the License Server and checks if the provided license key is valid. To use a proxy server, you must set the UseProxyServer, ProxyUser, ProxyDomain and ProxyPassword properties prior to calling this function.

C#: bool IsLicenseKeyRevoked (string webServiceUrl, string activationKey)

Parameters

webServiceUrl - URL to the QLM License Server.

activationKey - the license key to activate

IsLicenseKeyValid

Connects to the License Server and checks if the provided license key is valid. To use a proxy server, you must set the UseProxyServer, ProxyUser, ProxyDomain and ProxyPassword properties prior to calling this function.

C#: void IsLicenseKeyValid (string webServiceUrl, string activationKey, out string response)

Parameters

webServiceUrl - URL to the QLM License Server.

activationKey - the license key to activate

response - XML fragment containing the result of the call. The Xml fragment schema is as follows:

- <?xml version='1.0' encoding='UTF-8'?>
- <QuickLicenseManager>
- <result>The activation key is valid.</result>
- </QuickLicenseManager>

In the event of an error, the XML fragments returns:

- <?xml version='1.0' encoding='UTF-8'?>
- <QuickLicenseManager>
- <error>The activation key is not valid</error>
- </QuickLicenseManager>

Ping

Pings the License Server and returns the UTC date on the server.

C#: bool Ping(string webServiceUrl, out string response, out DateTime serverDate)

Parameters

webServiceUrl - URL to the QLM License Server.

response - the response from the server

serverDate - the date/time on the server in UTC.

If the function returns true, the server is accessible.

ReleaseLicense

Releases a license key over the internet. To use a proxy server, you must set the UseProxyServer, ProxyUser, ProxyDomain and ProxyPassword properties prior to calling this function.

You can control how many times a user can release a license with the following Server Properties:

- maxReleaseCount: The maximum number of times an end-user can release a license.
- maxReleasePeriodInDays: When counting the number of released licenses, only count the ones that have been released in the past "maxReleasePeriodInDays" days. For example, if you want to allow a user to release a license twice per month, set maxReleasePeriodInDays to 30 and maxReleaseCount to 2.
- maxReleasePerClient: When counting the number of released licenses for a given activation key, count only the ones associated to a specific client. By default, QLM counts all the released licenses for a given activation regardless of the client system.

C#: void ReleaseLicense (string webServiceUrl, string activationKey, string computerID, out string response)

Parameters

webServiceUrl - URL to the QLM License Server.

activationKey - the license key to activate

computerID - the unique computer identifier

response - XML fragment containing the result of the call. The Xml fragment schema is as follows:

- <?xml version='1.0' encoding='UTF-8'?>
- <QuickLicenseManager>
- <result>ActivationKey A162DCF05C30D371A2D0E0461040A0 has been released.</result>
- </QuickLicenseManager>

In the event of an error, the XML fragments returns:

- <?xml version='1.0' encoding='UTF-8'?>
- <QuickLicenseManager>
- <error>Details about the error</error>
- </QuickLicenseManager>

SetUserData

Sets the user data field associated to an activation key.

C#: bool SetUserData (string webServiceUrl, string activationKey, string userData, out string errorMessage)

Parameters

webServiceUrl - URL to the QLM License Server.

activationKey - the activation key.

userData - the user data to set

errorMessage - returned error message

If the function succeeds, the return value is true. If the function fails, the return value is false. The errorMessage contains details about the error.

SetUserDataInActivationLog

Sets the user data field associated to an activation key and a computer ID. This method should be used for multiple activations keys.

C#: bool SetUserDataInActivationLog (string webServiceUrl, string activationKey, string computerID, string userData, out string errorMessage)

Parameters

webServiceUrl - URL to the QLM License Server.

activationKey - the activation key.

computerID - the unique identifier of the system.

userData - the user data to set

errorMessage - returned error message

If the function succeeds, the return value is true. If the function fails, the return value is false. The errorMessage contains details about the error.

SubscribeToMailList

Subscribes or unsubscribes a user from the mail list. To use a proxy server, you must set the UseProxyServer, ProxyUser, ProxyDomain and ProxyPassword properties prior to calling this function.

C#: bool SubscribeToMailList (string webServiceUrl, string customerEmail, bool includeInMailList, out string response)

Http:

http://server/qlm/qlmservice.asmx/SubscribeToMailListHttp?is_email=user@cie.com&is_include=1

;

Parameters

webServiceUrl - URL to the QLM License Server.

customerEmail - email address of customer

includeInMailList - true to subscribe, false to unsubscribe

response - XML fragment containing the result of the call. The Xml fragment schema is as follows:

- <?xml version='1.0' encoding='UTF-8'?>
- <QuickLicenseManager>
- <result>Succesfully subscribed customer.</result>
- </QuickLicenseManager>

In the event of an error, the XML fragments returns:

- <?xml version='1.0' encoding='UTF-8'?>
- <QuickLicenseManager>
- <error>Details about the error</error>
- </QuickLicenseManager>

UpdateUser

Updates the data of an existing user. To use a proxy server, you must set the UseProxyServer, ProxyUser, ProxyDomain and ProxyPassword properties prior to calling this function.

C#: void UpdateUser(string webServiceUrl, string previousEmail, string customerName, string customerEmail, string customerPhone, string customerFax, string customerMobile, string customerCompany, string customerAddress1, string customerAddress2, string customerCity, string customerState, string customerZip, string customerCountry, string customerIP, string customerNotes, bool includeInMailList, out string response)

Parameters

webServiceUrl - URL to the QLM License Server
previousEmail - Email address of the existing user to update
customerName - Full Name
customerEmail - Email address
customerPhone - Phone number
customerFax - Fax number
customerMobile - Mobile phone number
customerCompany - Company name
customerAddress1 - Address 1
customerAddress2 - Address 2
customerCity - City
customerState - State
customerZip - Zip Code
customerCountry - Country
customerIP - IP Address
customerNotes - Notes
includeInMailList - Include in mail list
response - XML fragment containing the result of the call. The Xml fragment schema is as follows:

- <?xml version='1.0' encoding='UTF-8'?>
- <QuickLicenseManager>
- <result>Customer ABC was updated successfully."</result>
- </QuickLicenseManager>

In the event of an error, the XML fragments returns:

- <?xml version='1.0' encoding='UTF-8'?>
- <QuickLicenseManager>
- <error>Details about the error</error>
- </QuickLicenseManager>

License Server Management API

The QLM License Server provides an extended set of functions that you may want to use for managing your license keys or integrating licensing with any other internal process that you may have. This extended set of functions should typically not be part of your application but rather called from a server or a system that your end-user does not have access to.

Before calling any of the functions below, you need to set the `adminEncryptionKey` property of the `QlmLicense` object to the value specified in your Site Properties (Manage Keys tab / Site). If you are developing an application in .NET, it is highly recommended that you obfuscate your code, and more specifically that you encrypt sensitive strings such as the `adminEncryptionKey`.

CreateActivationKey

Creates an activation key over the internet. To use a proxy server, you must set the UseProxyServer, ProxyUser, ProxyDomain and ProxyPassword properties prior to calling this function.

Note that to call this function, you must update the web.config on the web server as follows:

```
<setting name="enableCreateActivationKey" serializeAs="String">
<value>True</value>
</setting>
```

C#: void CreateActivationKey(string webServiceUrl, string email, int features, int quantity, bool useMultipleActivationsKey, string qlmVersion, string vendor, string userData1, string affiliateID, out string response)

Parameters

webServiceUrl - URL to the QLM License Server.

email - email address to associate to the license key - may be empty

features - or'ed value of the features to enable

quantity - the number of licenses to embed in the key

useMultipleActivationsKey - if set to true and quantity > 1, one license key will be generated for all required licenses. The number of licenses will be embedded in the license key

qlmVersion - the version of the QLM Engine

vendor - the eCommerce vendor to use when generating the key

userData1 - user data to associate to this license

affiliateID - ID of affiliate

response - XML fragment containing the result of the call. The Xml fragment schema is as follows:

- <?xml version='1.0' encoding='UTF-8'?>
- <QuickLicenseManager>
- <keys>A062E-9D0CC-6DC80-0D6A0-E0701-000A0;A062E-9D0CC-6DC80-0D6A0-E0701-000A0</keys>
- </QuickLicenseManager>

In the event of an error, the XML fragments returns:

- <?xml version='1.0' encoding='UTF-8'?>
- <QuickLicenseManager>
- <error>Details about the error</error>
- </QuickLicenseManager>

CreateActivationKeyEx

Creates an activation key over the internet. To use a proxy server, you must set the UseProxyServer, ProxyUser, ProxyDomain and ProxyPassword properties prior to calling this function.

Note that to call this function, you must update the web.config on the web server as follows:

```
<setting name="enableCreateActivationKey" serializeAs="String">
<value>True</value>
</setting>
```

C#: void CreateActivationKeyEx(string webServiceUrl, string email, int [] features, int quantity, bool useMultipleActivationsKey, string qlmVersion, string vendor, string userData1, string affiliateID, out string response)

Parameters

webServiceUrl - URL to the QLM License Server.

email - email address to associate to the license key - may be empty

features - array of feature sets. each feature set is an or'ed value of the features to enable in the feature set

quantity - the number of licenses to embed in the key

useMultipleActivationsKey - if set to true and quantity > 1, one license key will be generated for all required licenses. The number of licenses will be embedded in the license key

qlmVersion - the version of the QLM Engine

vendor - the eCommerce vendor to use when generating the key

userData1 - user data to associate to this license

affiliateID - ID of affiliate

response - XML fragment containing the result of the call. The Xml fragment schema is as follows:

- <?xml version='1.0' encoding='UTF-8'?>
- <QuickLicenseManager>
- <keys>A062E-9D0CC-6DC80-0D6A0-E0701-000A0;A062E-9D0CC-6DC80-0D6A0-E0701-000A0</keys>
- </QuickLicenseManager>

In the event of an error, the XML fragments returns:

- <?xml version='1.0' encoding='UTF-8'?>
- <QuickLicenseManager>
- <error>Details about the error</error>
- </QuickLicenseManager>

CreateActivationKeyWithExpiryDate

Creates an activation key with an expiry date over the internet. To use a proxy server, you must set the UseProxyServer, ProxyUser, ProxyDomain and ProxyPassword properties prior to calling this function. Note that to call this function, you must update the web.config on the web server as follows:

```
<setting name="enableCreateActivationKey" serializeAs="String">
<value>True</value>
</setting>
```

C#: void CreateActivationKeyWithExpiryDate(string webServiceUrl, string email, int features, int quantity, bool useMultipleActivationsKey, string qlmVersion, string vendor, string userData1, string affiliateID, DateTime expiryDate, int expiryDuration, out string response)

Parameters

webServiceUrl - URL to the QLM License Server.
email - email address to associate to the license key - may be empty
features - or'ed value of the features to enable
quantity - the number of licenses to embed in the key
useMultipleActivationsKey - if set to true and quantity > 1, one license key will be generated for all required licenses. The number of licenses will be embedded in the license key
qlmVersion - the version of the QLM Engine
vendor - the eCommerce vendor to use when generating the key
userData1 - user data to associate to this license
affiliateID - ID of affiliate
expiryDate - Expiry date of the key
expiryDuration - Expiry duration of the key
response - XML fragment containing the result of the call. The Xml fragment schema is as follows:

- <?xml version='1.0' encoding='UTF-8'?>
- <QuickLicenseManager>
- <keys>A062E-9D0CC-6DC80-0D6A0-E0701-000A0;A062E-9D0CC-6DC80-0D6A0-E0701-000A0</keys>
- </QuickLicenseManager>

In the event of an error, the XML fragments returns:

- <?xml version='1.0' encoding='UTF-8'?>
- <QuickLicenseManager>
- <error>Details about the error</error>
- </QuickLicenseManager>

CreateActivationKeyWithExpiryDateEx

Creates an activation key with an expiry date over the internet. To use a proxy server, you must set the UseProxyServer, ProxyUser, ProxyDomain and ProxyPassword properties prior to calling this function. Note that to call this function, you must update the web.config on the web server as follows:

```
<setting name="enableCreateActivationKey" serializeAs="String">
<value>True</value>
</setting>
```

C#: void CreateActivationKeyWithExpiryDateEx(string webServiceUrl, string email, int [] features, int quantity, bool useMultipleActivationsKey, string qlmVersion, string vendor, string userData1, string affiliateID, DateTime expiryDate, int expiryDuration, out string response)

Parameters

webServiceUrl - URL to the QLM License Server.

email - email address to associate to the license key - may be empty

features - array of feature sets. each feature set is an or'ed value of the features to enable in the feature set

quantity - the number of licenses to embed in the key

useMultipleActivationsKey - if set to true and quantity > 1, one license key will be generated for all required licenses. The number of licenses will be embedded in the license key

qlmVersion - the version of the QLM Engine

vendor - the eCommerce vendor to use when generating the key

userData1 - user data to associate to this license

affiliateID - ID of affiliate

expiryDate - Expiry date of the key

expiryDuration - Expiry duration of the key

response - XML fragment containing the result of the call. The Xml fragment schema is as follows:

- <?xml version='1.0' encoding='UTF-8'?>
- <QuickLicenseManager>
- <keys>A062E-9D0CC-6DC80-0D6A0-E0701-000A0;A062E-9D0CC-6DC80-0D6A0-E0701-000A0</keys>
- </QuickLicenseManager>

In the event of an error, the XML fragments returns:

- <?xml version='1.0' encoding='UTF-8'?>
- <QuickLicenseManager>
- <error>Details about the error</error>
- </QuickLicenseManager>

CreateActivationKeyWithExpiryDateEx2

Creates an activation key with an expiry date over the internet. To use a proxy server, you must set the UseProxyServer, ProxyUser, ProxyDomain and ProxyPassword properties prior to calling this function. Note that to call this function, you must update the web.config on the web server as follows:

```
<setting name="enableCreateActivationKey" serializeAs="String">
<value>True</value>
</setting>
```

C#: void CreateActivationKeyWithExpiryDateEx2(string webServiceUrl, string email, string features, int numSeats, bool useMultipleActivationsKey, string qlmVersion, string vendor, string userData1, string affiliateID, DateTime expiryDate, int expiryDuration, out string response)

Parameters

webServiceUrl - URL to the QLM License Server.

email - email address to associate to the license key - may be empty

features - Semi comma separated list of feature sets and their corresponding enabled features.

Example: 0:1;1:2;2:3;3:6 - enables feature 1 in feature set 0, feature 2 in feature set 1, feature 1+2 (3) in feature set 4 and features 1+2+3 (6) in feature set 3.

quantity - the number of licenses to embed in the key

useMultipleActivationsKey - if set to true and quantity > 1, one license key will be generated for all required licenses. The number of licenses will be embedded in the license key

qlmVersion - the version of the QLM Engine

vendor - the eCommerce vendor to use when generating the key

userData1 - user data to associate to this license

affiliateID - ID of affiliate

expiryDate - Expiry date of the key

expiryDuration - Expiry duration of the key

response - XML fragment containing the result of the call. The Xml fragment schema is as follows:

- <?xml version='1.0' encoding='UTF-8'?>
- <QuickLicenseManager>
- <keys>A062E-9D0CC-6DC80-0D6A0-E0701-000A0;A062E-9D0CC-6DC80-0D6A0-E0701-000A0</keys>
- </QuickLicenseManager>

In the event of an error, the XML fragments returns:

- <?xml version='1.0' encoding='UTF-8'?>
- <QuickLicenseManager>
- <error>Details about the error</error>
- </QuickLicenseManager>

CreateActivationKeyWithExpiryDateEx3

Creates an activation key with an expiry date over the internet. To use a proxy server, you must set the UseProxyServer, ProxyUser, ProxyDomain and ProxyPassword properties prior to calling this function. Note that to call this function, you must update the web.config on the web server as follows:

```
<setting name="enableCreateActivationKey" serializeAs="String">
<value>True</value>
</setting>
```

C#: void CreateActivationKeyWithExpiryDateEx3(string webServiceUrl, string email, string features, int numKeysToCreate, int numSeats, int numFloatingSeats, bool useMultipleActivationsKey, string qlmVersion, string vendor, string userData1, string affiliateID, DateTime expiryDate, int expiryDuration, bool maintenance, bool generic, out string response)

Parameters

webServiceUrl - URL to the QLM License Server.

email - email address to associate to the license key - may be empty

features - Semi comma separated list of feature sets and their corresponding values.

Example: 0:1;1:2;2:3;3:6 - enables feature 1 in feature set 0, feature 2 in feature set 1, feature 1+2 (3) in feature set 4 and features 1+2+3 (6) in feature set 3.

numKeysToCreate - the number of licenses to create. Use this option to create large batches of license keys.

numSeats - the number of licenses to embed in the key. This controls how many activations are allowed per key.

numFloatingSeats - the number of floating seats for concurrent licensing (requires QLM Enterprise).

useMultipleActivationsKey - if set to true and quantity > 1, one license key will be generated for all required licenses. The number of licenses will be embedded in the license key

qlmVersion - the version of the QLM Engine

vendor - the eCommerce vendor to use when generating the key

userData1 - user data to associate to this license

affiliateID - ID of affiliate

expiryDate - Expiry date of the key

expiryDuration - Expiry duration of the key

maintenance - set to true to enable the maintenance plan for this license

generic - set to true to create a generic license key. Generic license keys are designed for enterprise customers who purchase hundreds of licenses and do not want to have to activate licenses on every single computer. They activate a single license and get back a Generic Computer Key. Then on every other computer in the organization, they use the Generic Computer Key.

response - XML fragment containing the result of the call. The Xml fragment schema is as follows:

- <?xml version='1.0' encoding='UTF-8'?>
- <QuickLicenseManager>
- <keys>A062E-9D0CC-6DC80-0D6A0-E0701-000A0;A062E-9D0CC-6DC80-0D6A0-E0701-000A0</keys>
- </QuickLicenseManager>

In the event of an error, the XML fragments returns:

- <?xml version='1.0' encoding='UTF-8'?>
- <QuickLicenseManager>
- <error>Details about the error</error>
- </QuickLicenseManager>

CreateActivationKeyWithExpiryDateEx4

Creates an activation key with an expiry date over the internet. To use a proxy server, you must set the UseProxyServer, ProxyUser, ProxyDomain and ProxyPassword properties prior to calling this function. Note that to call this function, you must update the web.config on the web server as follows:

```
<setting name="enableCreateActivationKey" serializeAs="String">
<value>True</value>
</setting>
```

C#: void CreateActivationKeyWithExpiryDateEx4(string webServiceUrl, string email, string features, int numKeysToCreate, int numSeats, int numFloatingSeats, bool useMultipleActivationsKey, string qlmVersion, string vendor, string userData1, string affiliateID, DateTime expiryDate, int expiryDuration, bool maintenance, bool generic, ELicenseModel licenseModel, out string response)

Parameters

webServiceUrl - URL to the QLM License Server.

email - email address to associate to the license key - may be empty

features - Semi comma separated list of feature sets and their corresponding values.

Example: 0:1;1:2;2:3;3:6 - enables feature 1 in feature set 0, feature 2 in feature set 1, feature 1+2 (3) in feature set 4 and features 1+2+3 (6) in feature set 3.

numKeysToCreate - the number of licenses to create. Use this option to create large batches of license keys.

numSeats - the number of licenses to embed in the key. This controls how many activations are allowed per key.

numFloatingSeats - the number of floating seats for concurrent licensing (requires QLM Enterprise).

useMultipleActivationsKey - if set to true and quantity > 1, one license key will be generated for all required licenses. The number of licenses will be embedded in the license key

qlmVersion - the version of the QLM Engine

vendor - the eCommerce vendor to use when generating the key

userData1 - user data to associate to this license

affiliateID - ID of affiliate

expiryDate - Expiry date of the key

expiryDuration - Expiry duration of the key

maintenance - set to true to enable the maintenance plan for this license

generic - set to true to create a generic license key. Generic license keys are designed for enterprise customers who purchase hundreds of licenses and do not want to have to activate licenses on every single computer. They activate a single license and get back a Generic Computer Key. Then on every other computer in the organization, they use the Generic Computer Key.

licenseModel - the license model associated to this license key. The license model can be Permanent | Trial | Subscription

response - XML fragment containing the result of the call. The Xml fragment schema is as follows:

- <?xml version='1.0' encoding='UTF-8'?>
- <QuickLicenseManager>
- <keys>A062E-9D0CC-6DC80-0D6A0-E0701-000A0;A062E-9D0CC-6DC80-0D6A0-E0701-000A0</keys>
- </QuickLicenseManager>

In the event of an error, the XML fragments returns:

- <?xml version='1.0' encoding='UTF-8'?>
- <QuickLicenseManager>
- <error>Details about the error</error>

- `</QuickLicenseManager>`

CreateActivationKeyWithExpiryDateEx5

Creates an activation key with an expiry date over the internet. To use a proxy server, you must set the UseProxyServer, ProxyUser, ProxyDomain and ProxyPassword properties prior to calling this function. Note that to call this function, you must update the web.config on the web server as follows:

```
<setting name="enableCreateActivationKey" serializeAs="String">
<value>True</value>
</setting>
```

C#: void CreateActivationKeyWithExpiryDateEx5(string webServiceUrl, string email, string features, int numKeysToCreate, int numSeats, int numFloatingSeats, bool useMultipleActivationsKey, string qlmVersion, string vendor, string userData1, string affiliateID, DateTime expiryDate, int expiryDuration, bool maintenance, bool generic, ELicenseModel licenseModel, string comment, out string response)

Parameters

webServiceUrl - URL to the QLM License Server.

email - email address to associate to the license key - may be empty

features - Semi comma separated list of feature sets and their corresponding values.

Example: 0:1;1:2;2:3;3:6 - enables feature 1 in feature set 0, feature 2 in feature set 1, feature 1+2 (3) in feature set 4 and features 1+2+3 (6) in feature set 3.

numKeysToCreate - the number of licenses to create. Use this option to create large batches of license keys.

numSeats - the number of licenses to embed in the key. This controls how many activations are allowed per key.

numFloatingSeats - the number of floating seats for concurrent licensing (requires QLM Enterprise).

useMultipleActivationsKey - if set to true and quantity > 1, one license key will be generated for all required licenses. The number of licenses will be embedded in the license key

qlmVersion - the version of the QLM Engine

vendor - the eCommerce vendor to use when generating the key

userData1 - user data to associate to this license

affiliateID - ID of affiliate

expiryDate - Expiry date of the key

expiryDuration - Expiry duration of the key

maintenance - set to true to enable the maintenance plan for this license

generic - set to true to create a generic license key. Generic license keys are designed for enterprise customers who purchase hundreds of licenses and do not want to have to activate licenses on every single computer. They activate a single license and get back a Generic Computer Key. Then on every other computer in the organization, they use the Generic Computer Key.

licenseModel - the license model associated to this license key. The license model can be Permanent | Trial | Subscription

comment - comment to be added to the license record

response - XML fragment containing the result of the call. The Xml fragment schema is as follows:

- <?xml version='1.0' encoding='UTF-8'?>
- <QuickLicenseManager>
- <keys>A062E-9D0CC-6DC80-0D6A0-E0701-000A0;A062E-9D0CC-6DC80-0D6A0-E0701-000A0</keys>
- </QuickLicenseManager>

In the event of an error, the XML fragments returns:

- <?xml version='1.0' encoding='UTF-8'?>
- <QuickLicenseManager>

- `<error>Details about the error</error>`
- `</QuickLicenseManager>`

CreateActivationKeyWithExpiryDateEx6

Creates an activation key with an expiry date over the internet. To use a proxy server, you must set the UseProxyServer, ProxyUser, ProxyDomain and ProxyPassword properties prior to calling this function. Note that to call this function, you must update the web.config on the web server as follows:

```
<setting name="enableCreateActivationKey" serializeAs="String">
<value>True</value>
</setting>
```

C#: void CreateActivationKeyWithExpiryDateEx6(string webServiceUrl, string email, string features, int numKeysToCreate, int numSeats, int numFloatingSeats, bool useMultipleActivationsKey, string qlmVersion, string vendor, string userData1, string affiliateID, DateTime expiryDate, int expiryDuration, bool maintenance, bool generic, ELicenseModel licenseModel, string comment, EOrderStatus orderStatus, out string response)

Parameters

webServiceUrl - URL to the QLM License Server.

email - email address to associate to the license key - may be empty

features - Semi comma separated list of feature sets and their corresponding values.

Example: 0:1;1:2;2:3;3:6 - enables feature 1 in feature set 0, feature 2 in feature set 1, feature 1+2 (3) in feature set 4 and features 1+2+3 (6) in feature set 3.

numKeysToCreate - the number of licenses to create. Use this option to create large batches of license keys.

numSeats - the number of licenses to embed in the key. This controls how many activations are allowed per key.

numFloatingSeats - the number of floating seats for concurrent licensing (requires QLM Enterprise).

useMultipleActivationsKey - if set to true and quantity > 1, one license key will be generated for all required licenses. The number of licenses will be embedded in the license key

qlmVersion - the version of the QLM Engine

vendor - the eCommerce vendor to use when generating the key

userData1 - user data to associate to this license

affiliateID - ID of affiliate

expiryDate - Expiry date of the key

expiryDuration - Expiry duration of the key

maintenance - set to true to enable the maintenance plan for this license

generic - set to true to create a generic license key. Generic license keys are designed for enterprise customers who purchase hundreds of licenses and do not want to have to activate licenses on every single computer. They activate a single license and get back a Generic Computer Key. Then on every other computer in the organization, they use the Generic Computer Key.

licenseModel - the license model associated to this license key. The license model can be Permanent | Trial | Subscription

comment - comment to be added to the license record

orderStatus - set the order status of the license to one of the allowed values: EInProgress | EComplete | EUpgraded | EReleased

response - XML fragment containing the result of the call. The Xml fragment schema is as follows:

- <?xml version='1.0' encoding='UTF-8'?>
- <QuickLicenseManager>
- <keys>A062E-9D0CC-6DC80-0D6A0-E0701-000A0;A062E-9D0CC-6DC80-0D6A0-E0701-000A0</keys>
- </QuickLicenseManager>

In the event of an error, the XML fragments returns:

- `<?xml version='1.0' encoding='UTF-8'?>`
- `<QuickLicenseManager>`
- `<error>Details about the error</error>`
- `</QuickLicenseManager>`

CreateActivationKeyWithExpiryDateEx7

Creates an activation key with an expiry date over the internet. To use a proxy server, you must set the UseProxyServer, ProxyUser, ProxyDomain and ProxyPassword properties prior to calling this function. Note that to call this function, you must update the web.config on the web server as follows:

```
<setting name="enableCreateActivationKey" serializeAs="String">
<value>True</value>
</setting>
```

C#: void CreateActivationKeyWithExpiryDateEx7(string webServiceUrl, string email, int[] features, int numKeysToCreate, int numSeats, int numFloatingSeats, bool useMultipleActivationsKey, string qlmVersion, string vendor, string userData1, string affiliateID, DateTime expiryDate, int expiryDuration, bool maintenance, bool generic, ELicenseModel licenseModel, string comment, EOrderStatus orderStatus, string productProperties, out string response)

Parameters

webServiceUrl - URL to the QLM License Server.

email - email address to associate to the license key - may be empty

features - features - 4 element array of features. Each element in the array represents a feature set and the value of each element is the OR'ed value of all the enabled features in this feature set.

 For example:

  int[] features=new features[4];

  features[0] = 1 + 2 + 4; // In feature set 1, Feature 1, Feature 2 and Feature 4 are enabled

  features[1] = 1 + 4; // In feature set 2, Feature 1 and Feature 4 are enabled

  features[2] = 2 + 4 + 8; // In feature set 3, Feature 2, Feature 4 and Feature 8 are enabled

  features[3] = 0; // In feature set 4, no features are enabled

numKeysToCreate - the number of licenses to create. Use this option to create large batches of license keys.

numSeats- the number of licenses to embed in the key. This controls how many activations are allowed per key.

numFloatingSeats- the number of floating seats for concurrent licensing (requires QLM Enterprise).

useMultipleActivationsKey - if set to true and quantity > 1, one license key will be generated for all required licenses. The number of licenses will be embedded in the license key

qlmVersion - the version of the QLM Engine

vendor - the eCommerce vendor to use when generating the key

userData1 - user data to associate to this license

affiliateID - ID of affiliate

expiryDate - Expiry date of the key

expiryDuration - Expiry duration of the key

maintenance - set to true to enable the maintenance plan for this license

generic - set to true to create a generic license key. Generic license keys are designed for enterprise customers who purchase hundreds of licenses and do not want to have to activate licenses on every single computer. They activate a single license and get back a Generic Computer Key. Then on every other computer in the organization, they use the Generic Computer Key.

licenseModel - the license model associated to this license key. The license model can be Permanent | Trial | Subscription

comment - comment to be added to the license record

orderStatus - set the order status of the license to one of the allowed values: EInProgress | EComplete | EUpgraded | EReleased

productProperties: xml string representing the product properties to set. The xml string can be created by calling `IProductProperties.Serialize()`.

response - XML fragment containing the result of the call. The Xml fragment schema is as follows:

- `<?xml version='1.0' encoding='UTF-8'?>`
- `<QuickLicenseManager>`
- `<keys>A062E-9D0CC-6DC80-0D6A0-E0701-000A0;A062E-9D0CC-6DC80-0D6A0-E0701-000A0</keys>`
- `</QuickLicenseManager>`

In the event of an error, the XML fragments returns:

- `<?xml version='1.0' encoding='UTF-8'?>`
- `<QuickLicenseManager>`
- `<error>Details about the error</error>`
- `</QuickLicenseManager>`

CreateOrder

Creates an order and an activation key with an expiry date over the internet. To use a proxy server, you must set the UseProxyServer, ProxyUser, ProxyDomain and ProxyPassword properties prior to calling this function.

Note that to call this function, you must update the web.config on the web server as follows:

```
<setting name="enableCreateOrder" serializeAs="String">
<value>True</value>
</setting>
```

C#: void CreateOrder(string webServiceUrl, string email, int features, int quantity, bool useMultipleActivationsKey, string qlmVersion, string vendor, string userData1, string affiliateID, DateTime expiryDate, int expiryDuration, string orderID, int orderStatus, out string response)

Parameters

webServiceUrl - URL to the QLM License Server.
email - email address to associate to the license key - may be empty
features - or'ed value of the features to enable
quantity - the number of licenses to embed in the key
useMultipleActivationsKey - if set to true and quantity > 1, one license key will be generated for all required licenses. The number of licenses will be embedded in the license key
qlmVersion - the version of the QLM Engine
vendor - the eCommerce vendor to use when generating the key
userData1 - user data to associate to this license
affiliateID - ID of affiliate
expiryDate - Expiry date of the key
expiryDuration - Expiry duration of the key
orderID - order ID
orderStatus - status of the order. Possible values are: None (0), In Progress (1), Completed (2).
response - XML fragment containing the result of the call. The Xml fragment schema is as follows:

- <?xml version='1.0' encoding='UTF-8'?>
- <QuickLicenseManager>
- <keys>A062E-9D0CC-6DC80-0D6A0-E0701-000A0;A062E-9D0CC-6DC80-0D6A0-E0701-000A0</keys>
- </QuickLicenseManager>

In the event of an error, the XML fragments returns:

- <?xml version='1.0' encoding='UTF-8'?>
- <QuickLicenseManager>
- <error>Details about the error</error>
- </QuickLicenseManager>

CreateOrderEx

Creates an order and an activation key with an expiry date over the internet. To use a proxy server, you must set the UseProxyServer, ProxyUser, ProxyDomain and ProxyPassword properties prior to calling this function.

Note that to call this function, you must update the web.config on the web server as follows:

```
<setting name="enableCreateOrder" serializeAs="String">  
<value>True</value>  
</setting>
```

C#: void CreateOrderEx(string webServiceUrl, string email, int [] features, int quantity, bool useMultipleActivationsKey, string qlmVersion, string vendor, string userData1, string affiliateID, DateTime expiryDate, int expiryDuration, string orderID, int orderStatus, out string response)

Parameters

webServiceUrl - URL to the QLM License Server.
email - email address to associate to the license key - may be empty
features - array of feature sets. each feature set is an or'ed value of the features to enable in the feature set
quantity - the number of licenses to embed in the key
useMultipleActivationsKey - if set to true and quantity > 1, one license key will be generated for all required licenses. The number of licenses will be embedded in the license key
qlmVersion - the version of the QLM Engine
vendor - the eCommerce vendor to use when generating the key
userData1 - user data to associate to this license
affiliateID - ID of affiliate
expiryDate - Expiry date of the key
expiryDuration - Expiry duration of the key
orderID - order ID
orderStatus - status of the order. Possible values are: None (0), In Progress (1), Completed (2).
response - XML fragment containing the result of the call. The Xml fragment schema is as follows:

- <?xml version='1.0' encoding='UTF-8'?>
- <QuickLicenseManager>
- <keys>A062E-9D0CC-6DC80-0D6A0-E0701-000A0;A062E-9D0CC-6DC80-0D6A0-E0701-000A0</keys>
- </QuickLicenseManager>

In the event of an error, the XML fragments returns:

- <?xml version='1.0' encoding='UTF-8'?>
- <QuickLicenseManager>
- <error>Details about the error</error>
- </QuickLicenseManager>

DeleteLicense

Deletes a license key over the internet. To use a proxy server, you must set the UseProxyServer, ProxyUser, ProxyDomain and ProxyPassword properties prior to calling this function.

C#: void DeleteLicense (string webServiceUrl, string activationKey, string computerID, bool multipleActivationsKey, bool historyTable, out string response)

Parameters

webServiceUrl - URL to the QLM License Server.

activationKey - the license key to delete

computerID - the computer ID to delete. The computerID is optional if multipleActivationsKey is false.

multipleActivationsKey - set to true if this key is a multiple activations key. The Delete operates then on the ActivationLog table. The computerID must be specified.

historyTable - set to true if you want to delete licenses from the history table where all released licenses are backed up.

response - XML fragment containing the result of the call. The Xml fragment schema is as follows:

- <?xml version='1.0' encoding='UTF-8'?>
- <QuickLicenseManager>
- <result>ActivationKey A162DCF05C30D371A2D0E0461040A0 has been deleted.</result>
- </QuickLicenseManager>

In the event of an error, the XML fragments returns:

- <?xml version='1.0' encoding='UTF-8'?>
- <QuickLicenseManager>
- <error>Details about the error</error>
- </QuickLicenseManager>

DownloadProducts

Downloads the list of products from the License Server over the internet. To use a proxy server, you must set the UseProxyServer, ProxyUser, ProxyDomain and ProxyPassword properties prior to calling this function.

C#: void DownloadProducts (string webServiceUrl, ref string dataSet, out string response)

Parameters

webServiceUrl - URL to the QLM License Server.

serverDate - returns the date the products were uploaded to the server.

dataSet - returned dataset containing products

response - XML fragment containing the result of the call. The Xml fragment schema is as follows:

- <?xml version='1.0' encoding='UTF-8'?>
- <QuickLicenseManager>
- <result>Succesfully downloaded products.</result>
- </QuickLicenseManager>

In the event of an error, the XML fragments returns:

- <?xml version='1.0' encoding='UTF-8'?>
- <QuickLicenseManager>
- <error>Details about the error</error>
- </QuickLicenseManager>

GetCustomersInfo

Retrieves information about a set of customers.

C#: DataSet GetCustomersInfo (string webServiceUrl, string fieldName, string fieldOperator, string fieldValue, out string response)

Parameters

webServiceUrl - URL to the QLM License Server.

field - field from the Customers table. Typical fields are: Email, FullName, CustomerID

fieldOperator - a valid SQL operator such as: =, like, <

fieldValue - value of the field to match

response - XML fragment containing the result of the call. The Xml fragment schema is as follows:

- <?xml version='1.0' encoding='UTF-8'?>
- <QuickLicenseManager>
- <result>Successfully executed query ...</result>
- </QuickLicenseManager>

In the event of an error, the XML fragments returns:

- <?xml version='1.0' encoding='UTF-8'?>
- <QuickLicenseManager>
- <error>Details about the error</error>
- </QuickLicenseManager>

Example

- DataSet ds = license.GetCustomersInfo (webServiceUrl, "email", "=", "customer@mail.com", out response);

GetCustomersInfoEx

Retrieves information about a set of customers.

This function is similar

C#: IQlmCustomerInfo [] GetCustomersInfoEx (string webServiceUrl, string fieldName, string fieldOperator, string fieldValue)

Parameters

webServiceUrl - URL to the QLM License Server.

field - field from the Customers table. Typical fields are: Email, FullName, CustomerID

fieldOperator - a valid SQL operator such as: =, like, <>

fieldValue - value of the field to match

C# Example

- IQlmCustomerInfo[] customers = license.GetCustomersInfo (webServiceUrl, "email", "=", "customer@mail.com");

C++ Example

// Get all the customer records

```
SAFEARRAY * sa = qlmLicense->GetCustomersInfoEx (webServiceUrl  
_bstr_t(""),_bstr_t(""),_bstr_t(""));IQlmCustomerInfo *ci;
```

```
LONG numRecords = 0;
```

```
SafeArrayGetUBound (sa, 1, &numRecords);
```

```
for (LONG i=0; i < numRecords; i++)
```

```
{
```

```
SafeArrayGetElement (sa, &i, &ci);
```

```
}
```

```
// Destroy the safe array when done
```

```
SafeArrayDestroy (sa);
```

GetDataSet

Gets a data set in xml format that meets the criteria specified in the filter.

C#: void GetDataSet (string webServiceUrl, string filter, ref string dataSet, out string response)

Parameters

webServiceUrl - URL to the QLM License Server.

filter - SQL filter to determine which records to return. Use a where clause sql syntax, example: ActivationKey='AAAA'. Note that if the filter contains an Activation Key or a Computer Key, you must strip out the dashes in the license key. License keys in the database are stored without dashes.

dataSet - returned dataset containing license key records that match criteria

response - XML fragment containing the result of the call. The Xml fragment schema is as follows:

- <?xml version='1.0' encoding='UTF-8'?>
- <QuickLicenseManager>
- <result>Successfully executed query using filter...</result>
- </QuickLicenseManager>

In the event of an error, the XML fragments returns:

- <?xml version='1.0' encoding='UTF-8'?>
- <QuickLicenseManager>
- <error>Details about the error</error>
- </QuickLicenseManager>

GetDataSetEx

Gets a data set in xml format that meets the criteria specified in the filter.

C#: void GetDataSetEx (string webServiceUrl, string table, string filter, ref string dataSet, out string response)

Parameters

webServiceUrl - URL to the QLM License Server.

table- specify the table to query. The value can be: LicenseKeys or LicenseKeysHistory.

LicenseKeys is the table where all license keys are recorded. LicenseKeysHistory contains all the released license keys.

filter - SQL filter to determine which records to return. Use a where clause sql syntax, example: ActivationKey='AAAA'. Note that if the filter contains an Activation Key or a Computer Key, you must strip out the dashes in the license key. License keys in the database are stored without dashes.

dataSet - returned dataset containing license key records that match criteria

response - XML fragment containing the result of the call. The Xml fragment schema is as follows:

- <?xml version='1.0' encoding='UTF-8'?>
- <QuickLicenseManager>
- <result>Successfully executed query using filter...</result>
- </QuickLicenseManager>

In the event of an error, the XML fragments returns:

- <?xml version='1.0' encoding='UTF-8'?>
- <QuickLicenseManager>
- <error>Details about the error</error>
- </QuickLicenseManager>

RenewSubscription

Connects to the License Server and renews a subscription.

When a subscription is renewed, each activated license is automatically reactivated on the server and a new computer bound key is generated with a new expiry date. When customers reactivate their license, they receive the new computer bound key with the new expiry date, thus extending their subscription period.

C#: bool RenewSubscription (string webServiceUrl, string activationKey, DateTime expiryDate, out string errorMessage)

webServiceUrl - URL to the QLM License Server.

activationKey- activation key to extend

expiryDate - Expiry date of the subscription

errorMessage - Error message if the operation failed.

Return

True if the subscription renewal was successful.

SetMaintenancePlanRenewalDate

Sets the maintenance plan renewal date.

C#: bool SetMaintenancePlanRenewalDate (string webServiceUrl, string activationKey, DateTime maintenancePlanRenewalDate, out string errorMessage)

Parameters

webServiceUrl - URL to the QLM License Server.

activationKey - the activation key of the record.

maintenancePlanRenewalDate - the renewal date of the maintenance plan

errorMessage - returned error message

It is recommended to send a UTC date.

If the function succeeds, the return value is true. If the function fails, the return value is false. The errorMessage contains details about the error.

UpdateActivationLogInfo

Updates the data associated with a license key. To use a proxy server, you must set the UseProxyServer, ProxyUser, ProxyDomain and ProxyPassword properties prior to calling this function. The ActivationLog table is used when multiple licenses are issued from a single ActivationKey. In this case, the data associated with each activated computer is stored in the ActivationLog table instead of the LicenseKeys table. Therefore, to update data in the ActivationLog table, you need to specify which computer to update. The computerID, computerKey and computerName arguments can be specified to identify the computer. At least one of these arguments must be specified.

The ActivationLog table contains the following updatable fields:

- ComputerKey, ComputerName, ComputerID, ActivationDate, LastAccessedDate, ActivationCount

C#: bool UpdateActivationLogInfo (string webServiceUrl, string activationKey, string computerID, string computerKey, string computerName, string licenseData, out string response)

Parameters

webServiceUrl - URL to the QLM License Server.

activationKey- the license key to update

computerID- the ID the computer to update

computerKey- the computer bound key to update

computerName- the computer name to update

licenseData - XML fragment containing the fields to update. The XML fragment should be of the form:

- <licenseArguments
 - field1=" 'value' "
 - field2=" 'value' "
- </licenseArguments>
- where field1 is the name of a field in the LicenseKeys table. For fields of type date, you should use the following date/time format: yyyy-MM-dd HH:mm:ss
- Example:
- <licenseArguments
 - ComputerName= " 'my pc' "
 - UserData1=" 'my user data' "
- </licenseArguments>

response - XML fragment containing the result of the call. The Xml fragment schema is as follows:

- <?xml version='1.0' encoding='UTF-8'?>
- <QuickLicenseManager>
- <result>Successfully updated license information for ActivationKey=XYZ.</result>
- </QuickLicenseManager>

In the event of an error, the XML fragments returns:

- <?xml version='1.0' encoding='UTF-8'?>
- <QuickLicenseManager>
- <error>Details about the error</error>
- </QuickLicenseManager>

UpdateLicenseInfo

Updates the data associated with a license key. To use a proxy server, you must set the UseProxyServer, ProxyUser, ProxyDomain and ProxyPassword properties prior to calling this function.

C#: bool UpdateLicenseInfo (string webServiceUrl, string activationKey, string licenseData, out string response)

Parameters

webServiceUrl - URL to the QLM License Server.

activationKey- the license key to update

licenseData - XML fragment containing the fields to update. The XML fragment should be of the form:

- <licenseArguments
 - field1=" 'value'
 - field2=" 'value' "
- />
- where field1 is the name of a field in the LicenseKeys table. For fields of type date, you should use the following date/time format: yyyy-MM-dd HH:mm:ss
- Example:
- <licenseArguments
 - Comment= " 'my comment' "
 - UserData1=" 'my user data' "
 - OrderDate=" '2008-3-12 21:14:58' "
- />

response - XML fragment containing the result of the call. The Xml fragment schema is as follows:

- <?xml version='1.0' encoding='UTF-8'?>
- <QuickLicenseManager>
- <result>Successfully updated license information for ActivationKey=XYZ.</result>
- </QuickLicenseManager>

In the event of an error, the XML fragments returns:

- <?xml version='1.0' encoding='UTF-8'?>
- <QuickLicenseManager>
- <error>Details about the error</error>
- </QuickLicenseManager>

UpdateLicenseKey

Updates a license key with another license key. To use a proxy server, you must set the UseProxyServer, ProxyUser, ProxyDomain and ProxyPassword properties prior to calling this function.

C#: bool UpdateLicenseKey (string webServiceUrl, string currentKey, string newKey, out string response)

Parameters

webServiceUrl - URL to the QLM License Server.

currentKey - the current license key

newKey - the new license key

response - XML fragment containing the result of the call. The Xml fragment schema is as follows:

- <?xml version='1.0' encoding='UTF-8'?>
- <QuickLicenseManager>
- <result>Successfully updated license information for ActivationKey=XYZ.</result>
- </QuickLicenseManager>

In the event of an error, the XML fragments returns:

- <?xml version='1.0' encoding='UTF-8'?>
- <QuickLicenseManager>
- <error>Details about the error</error>
- </QuickLicenseManager>

UpdateOrderStatus

Connects to the License Server and updates the status of an order. To use a proxy server, you must set the UseProxyServer, ProxyUser, ProxyDomain and ProxyPassword properties prior to calling this function.

C#: bool UpdateOrderStatus (string webServiceUrl, string orderID, int orderStatus, out string response)

Parameters

webServiceUrl - URL to the QLM License Server.

orderID - ID of the order

orderStatus - status of the order to set

response - XML fragment containing the result of the call. The Xml fragment schema is as follows:

- <?xml version='1.0' encoding='UTF-8'?>
- <QuickLicenseManager>
- <result>Order status updated successfully.</result>
- </QuickLicenseManager>

In the event of an error, the XML fragments returns:

- <?xml version='1.0' encoding='UTF-8'?>
- <QuickLicenseManager>
- <error>The order id is not valid</error>
- </QuickLicenseManager>

Return

Boolean indicating whether the operation was successful.

Example:

```
QlmLicense license = new QlmLicense ();  
license.DefineProduct(1, "Demo", 1, 0, "DemoKey",  
"{24EAA3C1-3DD7-40E0-AEA3-D20AA17A6005}");  
string response = string.Empty;  
bool stat = license.UpdateOrderStatus  
("https://qlm3.net/qlmdemov16/QlmLicenseServer/qlmservice.asmx", 1234, 2, out response);
```

UpgradeLicense

Connects to the License Server and upgrades a license. You can upgrade the following data associated to a license:

- Features associated to a license
- Expiry date of the license
- Duration of the license
- Major and Minor version of the product
- The version of the QLM Engine used to generate the license key

When a license is upgraded, a new license key is generated and replaces the existing one. The old license is copied to the released licenses table. To use a proxy server, you must set the UseProxyServer, ProxyUser, ProxyDomain and ProxyPassword properties prior to calling this function.

C#: bool UpgradeLicense (string webServiceUrl, string activationKey, int productID, int majorVersion, int minorVersion, string qlmVersion, int[] features, DateTime dtExpiry, int expiryDuration, out string response)

Http:

http://server/qlm/qlmservice.asmx/UpgradeLicense?is_avkey=<activationKey>&is_productid=<pid>&is_majorversion=<majorVersion>&is_minorversion=<minorVersion>&is_expddate=<yyyy-mm-dd>&is_expduration=<expiry duration>

Example:

http://server/qlm/qlmservice.asmxUpgradeLicense?is_avkey=B0739A30F960FAA0FA3045D05600000000&is_productid=1&is_majorversion=1&is_minorversion=0&is_expddate=2008-06-01

Parameters

webServiceUrl - URL to the QLM License Server.

activationKey- activation key to update

productID - ID of the product

majorVersion - New major version of the product

minorVersion - New minor version of the product

qlmVersion - Version of the QLM Engine to use.

features - An array of feature sets specifying the features that should be enabled in the created key. Each feature has a unique feature set and ID associated to it. To combine features, perform a bitwise OR operation on the required features.

dtExpiry - Expiry date of the license key

expiryDuration - Expiry duration of the license key

response - XML fragment containing the result of the call. The Xml fragment schema is as follows:

- <?xml version='1.0' encoding='UTF-8'?>
- <QuickLicenseManager>
- <result>The license was upgraded...</result>
- </QuickLicenseManager>

In the event of an error, the XML fragments returns:

- <?xml version='1.0' encoding='UTF-8'?>
- <QuickLicenseManager>
- <error>The activation key is not valid</error>
- </QuickLicenseManager>

Return

True if the upgrade was successful.

UpgradeLicenseEx

Connects to the License Server and upgrades a license. You can upgrade the following data associated to a license:

- Features associated to a license
- Expiry date of the license
- Duration of the license
- Major and Minor version of the product
- The version of the QLM Engine used to generate the license key

When a license is upgraded, a new license key is generated and replaces the existing one. The old license is copied to the released licenses table. To use a proxy server, you must set the UseProxyServer, ProxyUser, ProxyDomain and ProxyPassword properties prior to calling this function.

C#: bool UpgradeLicenseEx (string webServiceUrl, string activationKey, int productID, int majorVersion, int minorVersion, string qlmVersion, string features, DateTime dtExpiry, int expiryDuration, string comment, out newActivationKey, out string response)

Parameters

webServiceUrl - URL to the QLM License Server.

activationKey - activation key to update

productID - ID of the product

majorVersion - New major version of the product

minorVersion - New minor version of the product

qlmVersion - Version of the QLM Engine to use.

features - A semi comma separated list of enabled feature sets. For example, to enable features 001 and 002 in feature set 0 and features 004 and 008 in feature set 2, you use: "0:3;2:12"

dtExpiry - Expiry date of the license key

expiryDuration - Expiry duration of the license key

comment - A comment to add to the license record

newActivationKey - the newly generated activation key (out argument)

response - XML fragment containing the result of the call. The Xml fragment schema is as follows:

- <?xml version='1.0' encoding='UTF-8'?>
- <QuickLicenseManager>
- <result>The license was upgraded...</result>
- </QuickLicenseManager>

In the event of an error, the XML fragments returns:

- <?xml version='1.0' encoding='UTF-8'?>
- <QuickLicenseManager>
- <error>The activation key is not valid</error>
- </QuickLicenseManager>

Return

True if the upgrade was successful.

UploadProducts

Uploads the list of products to the License Server over the internet. To use a proxy server, you must set the UseProxyServer, ProxyUser, ProxyDomain and ProxyPassword properties prior to calling this function.

C#: void UploadProducts (string webServiceUrl, string productsXml, DateTime updatedUtcDate, out string response)

Parameters

webServiceUrl - URL to the QLM License Server.

productsXml - >Xml file containing the list of products. The format of the XML file is identical to the products.xml file stored on the client side. For example:

```
<?xml version='1.0' encoding='UTF-8'?>
<LICENSES>
<PRODUCTS>
<PRODUCT Name="Demo" ID="1" Major="2" Minor="0" Key="DemoKey"
GUID="{AB932603-7336-4DA4-90C1-843C4146E388}" ReleaseDate="2008-12-11"
Features="" LatestVersion="2.0" LatestVersionUrl="" LatestVersionNotes="" />
<PRODUCT Name="Demo" ID="1" Major="1" Minor="0" Key="DemoKey"
GUID="{24EAA3C1-3DD7-40E0-AEA3-D20AA17A6005}"
ReleaseDate="2007-12-01" Features="0:1:F1;0:2:F2;0:4:F3;3:1:D1;" LatestVersion="1.1"
LatestVersionUrl="http://yourserver/setup.exe" LatestVersionNotes="In this field, insert
comments that describe the latest version of your product.
```

This information can be retrieved by your application when checking for updates and displayed to the end user." />

```
</PRODUCTS>
```

```
</LICENSES>
```

updatedUtcDate - UTC Date at which the products were last updated.

response - XML fragment containing the result of the call. The Xml fragment schema is as follows:

```
<?xml version='1.0' encoding='UTF-8'?>
<QuickLicenseManager>
<result>Successfully downloaded products.</result>
</QuickLicenseManager>
```

In the event of an error, the XML fragments returns:

```
<?xml version='1.0' encoding='UTF-8'?>
<QuickLicenseManager>
<error>Details about the error</error>
</QuickLicenseManager>
```

Client Side API

QLM Pro provides a set of client side functions that you can use in your application or in any other order processing application that you may have. Client side functions do not communicate with the QLM License Server.

Before calling any of the functions below, you need to call the DefineProduct function and set the PublicKey property.

Functions that create license keys such as CreateLicenseKey should not typically be used from within your application. They should be used from a server or a system that the end user does not have access to. In order to call any of the functions that create license keys, you must set the PrivateKey property of the QlmLicense object.

The Public and Private keys for your product can be found in the Define Products tab / Keys tab.

If you are developing an application in .NET, it is highly recommended that you obfuscate your code, and more specifically that you encrypt sensitive strings such as the PrivateKey.

BackwardCompatible

Set this property to true to allow validation of keys prior to the latest version of the QLM engine.

C++: VARIANT_BOOL BackwardCompatible

C#: bool BackwardCompatible

CreateLicenseKey

Creates a non-computer bound license key. If the ExpiryDate is NULL and the ExpiryDuration is -1, the license key is a permanent non-evaluation license key.

Prior to calling this function, you must call DefineProduct and set the PrivateKey property. Note that including the PrivateKey in your code is not recommended. Creation of license keys should not typically be done from within the application but rather from a server that the user does not have access to.

C++: `_bstr_t CreateLicenseKey (DATE ExpiryDate, int ExpiryDuration)`

C#: `string CreateLicenseKey (System.DateTime ExpiryDate, int ExpiryDuration)`

Parameters

ExpiryDate - The date when the license will expire. Use NULL if you do not want to specify an expiry date.

ExpiryDuration - The duration of the evaluation period in days. Use -1 if you do not want to specify a duration.

Return

A non-computer bound license key.

CreateLicenseKeyEx

Creates a computer bound license key.

Prior to calling this function, you must call DefineProduct and set the PrivateKey property. Note that including the PrivateKey in your code is not recommended. Creation of license keys should not typically be done from within the application but rather from a server that the user does not have access to.

C++: `_bstr_t CreateLicenseKeyEx (ELicenseType LicenseType, BSTR MachineID)`

C#: `string CreateLicenseKeyEx (ELicenseType LicenseType, string MachineID)`

Parameters

LicenseType - Specify the type of license to generate. See the definition of LicenseType below.

MachineID - A unique identifier for the machine. If you specify a *ComputerName* as the LicenseType, this argument must be the Computer Name. If you specify *User Defined* as the LicenseType, this argument can be anything you want. When validating the license key in your code, you will need to provide the same value to the ValidateLicenseEx function.

Return

A computer bound license key.

CreateLicenseKeyEx2

Creates a computer bound license key that has an expiry date and a number of licenses.

Prior to calling this function, you must call DefineProduct and set the PrivateKey property. Note that including the PrivateKey in your code is not recommended. Creation of license keys should not typically be done from within the application but rather from a server that the user does not have access to.

C++: `_bstr_t CreateLicenseKeyEx2 (DATE ExpiryDate, int ExpiryDuration, int NumberOfLicenses, ELicenseType LicenseType, BSTR MachineID)`

C#: `string CreateLicenseKeyEx2 (System.DateTime ExpiryDate, int ExpiryDuration, int NumberOfLicenses, ELicenseType LicenseType, string MachineID)`

Parameters

ExpiryDate - The date when the license will expire. Use NULL if you do not want to specify an expiry date.

ExpiryDuration - The duration of the evaluation period in days. Use -1 if you do not want to specify a duration.

NumberOfLicenses - The number of licenses associated with the key. Use 1 if you do not want to use single activation licensing.

LicenseType - Specify the type of license to generate. See the definition of LicenseType below.

MachineID - A unique identifier for the machine. If you specify a *ComputerName* as the LicenseType, this argument must be the Computer Name. If you specify *User Defined* as the LicenseType, this argument can be anything you want. When validating the license key in your code, you will need to provide the same value to the ValidateLicenseEx function. To create a key that is not computer bound, set this argument to NULL and set the LicenseType to Generic.

Return

A computer bound license key.

CreateLicenseKeyEx3

Creates a computer bound license key that has an expiry date, a number of licenses and a specific set of features that are enabled.

Prior to calling this function, you must call DefineProduct and set the PrivateKey property. Note that including the PrivateKey in your code is not recommended. Creation of license keys should not typically be done from within the application but rather from a server that the user does not have access to.

C++: `_bstr_t CreateLicenseKeyEx3 (DATE expiryDate, int expiryDuration, int numberOfLicenses, ELicenseType licenseType, BSTR machineID, in features)`

C#: `string CreateLicenseKeyEx3 (System.DateTime expiryDate, int expiryDuration, int numberOfLicenses, ELicenseType licenseType, string machineID, int features)`

Parameters

ExpiryDate - The date when the license will expire. Use NULL if you do not want to specify an expiry date.

ExpiryDuration - The duration of the evaluation period in days. Use -1 if you do not want to specify a duration.

NumberOfLicenses - The number of licenses associated with the key. Use 1 if you do not want to use single activation licensing.

LicenseType - Specify the type of license to generate. See the definition of LicenseType below.

MachineID - A unique identifier for the machine. If you specify a *ComputerName* as the LicenseType, this argument must be the Computer Name. If you specify *User Defined* as the LicenseType, this argument can be anything you want. When validating the license key in your code, you will need to provide the same value to the ValidateLicenseEx function. To create a key that is not computer bound, set this argument to NULL and set the LicenseType to Generic.

Features - A value specifying the features that should be enabled in the created key. Each feature has a unique ID associated to it. To combine features, perform a bitwise OR operation on the required features.

Return

A computer bound license key.

CreateLicenseKeyEx4

Creates a computer bound license key that has an expiry date, a number of licenses and a specific set of features that are enabled.

Prior to calling this function, you must call DefineProduct and set the PrivateKey property. Note that including the PrivateKey in your code is not recommended. Creation of license keys should not typically be done from within the application but rather from a server that the user does not have access to.

C++: `_bstr_t CreateLicenseKeyEx4 (DATE expiryDate, int expiryDuration, int numberOfLicenses, ELicenseType licenseType, BSTR machineID, int[] features)`

C#: `string CreateLicenseKeyEx4 (System.DateTime expiryDate, int expiryDuration, int numberOfLicenses, ELicenseType licenseType, string machineID, SAFEARRAY *Features)`

Parameters

ExpiryDate - The date when the license will expire. Use NULL if you do not want to specify an expiry date.

ExpiryDuration - The duration of the evaluation period in days. Use -1 if you do not want to specify a duration.

NumberOfLicenses - The number of licenses associated with the key. Use 1 if you do not want to use single activation licensing.

LicenseType - Specify the type of license to generate. See the definition of LicenseType below.

MachineID - A unique identifier for the machine. If you specify a *ComputerName* as the LicenseType, this argument must be the Computer Name. If you specify *User Defined* as the LicenseType, this argument can be anything you want. When validating the license key in your code, you will need to provide the same value to the ValidateLicenseEx function. To create a key that is not computer bound, set this argument to NULL and set the LicenseType to Generic.

Features - An array of feature sets. Each feature set is a value specifying the features that should be enabled in the created key. The value of the feature set is the or'ed value of all the features to be enabled in the set. To combine features, perform a bitwise OR operation on the required features.

Return

A computer bound license key.

CreateLicenseKeyEx5

Creates a computer bound license key that has an expiry date, a number of licenses and a specific set of features that are enabled. This API is functionally identical to CreateLicenseKeyEx4. It was created to accomodate programming languages such as VB6 that cannot handle the array data type used in CreateLicenseKeyEx4.

Prior to calling this function, you must call DefineProduct and set the PrivateKey property. Note that including the PrivateKey in your code is not recommended. Creation of license keys should not typically be done from within the application but rather from a server that the user does not have access to.

C++: `_bstr_t CreateLicenseKeyEx5 (DATE expiryDate, int expiryDuration, int numberOfLicenses, ELicenseType licenseType, BSTR machineID, BSTR features)`

C#: `string CreateLicenseKeyEx5 (System.DateTime expiryDate, int expiryDuration, int numberOfLicenses, ELicenseType licenseType, string machineID, string Features)`

Parameters

ExpiryDate - The date when the license will expire. Use NULL if you do not want to specify an expiry date.

ExpiryDuration - The duration of the evaluation period in days. Use -1 if you do not want to specify a duration.

NumberOfLicenses - The number of licenses associated with the key. Use 1 if you do not want to use single activation licensing.

LicenseType - Specify the type of license to generate. See the definition of LicenseType below.

MachineID - A unique identifier for the machine. If you specify a *ComputerName* as the LicenseType, this argument must be the Computer Name. If you specify *User Defined* as the LicenseType, this argument can be anything you want. When validating the license key in your code, you will need to provide the same value to the ValidateLicenseEx function. To create a key that is not computer bound, set this argument to NULL and set the LicenseType to Generic.

Features - A set of features to be enabled using the following syntax:

`<featureSet>:<featureValue>;<featureSet>:<featureValue>`

Example: "0:8;1:2;3:14" - Enables: feature id 8 in feature set 0, feature id 2 in feature set 1 and feature ids 2, 4, 8 (2 + 4 + 8 = 14) in feature set 3.

To combine features, perform a bitwise OR operation on the required features.

Return

A computer bound license key.

DaysLeft

Returns the number of days left before the evaluation expires. You must call `ValidateLicense` prior to calling this function.

C++: `int DaysLeft`

C#: `int DaysLeft ()`

Return

Number of days left before the evaluation expires.

DefineProduct

The DefineProduct method initializes basic information required to validate license keys. You must call this function prior to calling any other function.

C++:

```
VARIANT_BOOL DefineProduct ( int ProductID, BSTR ProductName, int MajorVersion, int  
MinorVersion, BSTR EncryptionKey, BSTR PersistenceKey )
```

C#:

```
bool DefineProduct ( int ProductID, string ProductName, int MajorVersion, int MinorVersion, string  
EncryptionKey, string PersistenceKey )
```

Parameters

ProductID

: ID of the product as generated by Quick License Manager

ProductName

: Name of the product

MajorVersion

: Major version of the product (maximum 2 digits)

MinorVersion

: Minor version of the product (maximum 2 digits)

Encryption

Key: string used to encrypt the license key.

PersistenceKey

: GUID associated with the product and automatically generated by Quick License Manager for each product. The evaluation information of the product is stored at runtime in the registry under HKCR\CLSID\<GUID>.

DeleteKeys

Deletes the license keys stored on the end user system with the StoreKeys API. To store keys, use the StoreKeys API. To read the stored keys, use the ReadKeys API.

C#: void DeleteKeys ()

DeleteKeysEx

Deletes license keys, proxy settings and floating license settings stored on the end user system with the StoreKeys API. To store keys, use the StoreKeys API. To read the stored keys, use the ReadKeys API.

C#: void DeleteKeys (bool deleteActivationKey, bool deleteComputerKey, bool deleteProxySettings, bool deleteFloatingSettings, out string errorMessage))

Parameters

deleteActivationKey - If true, deletes the Activation Key.

deleteComputerKey - If true, deletes the Computer Key.

deleteProxySettings - If true, deletes Proxy Settings.

deleteFloatingSettings - If true, deletes Floating License settings.

errorMessage - errorMessage returned if some operations failed.

Duration

Returns the duration in days of the evaluation key. You must call `ValidateLicense` prior to calling this function.

C++: `int Duration`

C#: `int Duration ()`

Return

Duration of the evaluation key.

ELicenseStatus

Enum of all possible values of the license key status. Note that the status can consist of a combination of these values:

EKeyPermanent : The license key is valid and it is a permanent license key.

EKeyInvalid : The license key is invalid. It was not decoded successfully.

EKeyDemo : The license key is an evaluation key.

EKeyProductInvalid : The product ID of the license key does not correspond to the expected Product ID.

EKeyVersionInvalid : The Major or Minor version of the license key does not correspond to the expected Major or Minor version.

EKeyExpired : The license key has expired.

EKeyTampered

: The license key was tampered typically indicating that the user is attempting to set the date back to run the software.

EKeyMachineInvalid

: If you are using computer bound license keys, an *EKeyMachineInvalid* status indicates that the license key that was validated does not match the computer to which the license key was bound.

EKeyNeedsActivation

: This flag indicates that the license key is an activation key. If you detect an activation key, you should not enable your application. You should just allow the user to activate his license. Once the license is activated, a computer bound key is issued. Once you detect a valid computer bound key, you can enable your application.

ELicenseType

Enum of all possible types of license keys.

Activation : The license key is a key that requires activation.

Evaluation (obsolete) : The license key is an evaluation key.

ComputerName : The license key is bound to the name of the computer.

Generic (previously PermanentGeneric) : The license key is permanent and not bound to a computer.

UserDefined

: The license key is bound to the computer based on a user defined unique identifier.

EvaluationPerUser

Set this property to true to store evaluation information per user. The default value is true. If set to false, evaluation information is stored at the machine level. Note that you need to make sure the current user has the required privileges to store evaluation information at the machine level under `HKEY_LOCAL_MACHINE\Software\Classes\CLSID\<GUID>`.

Evaluation information consists of the installation date of your software as well as the last time your software ran.

C++: `VARIANT_BOOL EvaluationPerUser`

C#: `bool EvaluationPerUser`

ExpiryDate

Returns the expiry date of the evaluation key. You must call `ValidateLicense` prior to calling this function.

C++: `DATE ExpiryDate`

C#: `System.DateTime ExpiryDate()`

Return

Expiry date of the evaluation key.

FavorMachineLevelLicenseKey

This property affects the ReadKeys API.

If a license key is stored both at the machine level and user level, QLM will use the machine level key if this attribute is set to true. The default value is false.

For more details, read the help on how StoreKeys works.

Features

Returns an array of all the feature sets associated with the license key. Within a feature set (each element of the array), if several features are associated to a license key, the returned value is a bitwise OR of these features.

This function must be called after a call to `ValidateLicense` or `ValidateLicenseEx`.

C++: `int *Features`

C#:/STRONG> `int [] Features`

GetStatus

Returns the last status. See ELicenseStatus for possible values.

You must always call this function after calling ValidateLicense or ValidateLicenseEx to get the result of the validation.

C++: int GetStatus

C#: int GetStatus ()

Return

Last status

IsEvaluation

Returns whether the current license key is an evaluation key. You must call ValidateLicense prior to calling IsEvaluation.

C++: VARIANT_BOOL IsEvaluation

C#: bool IsEvaluation ()

Return

Boolean indicating if the license key is an evaluation key.

IsFeatureEnabled

Returns whether the specified feature is enabled in this license key. This function is now obsolete and has been superseded by IsFeatureEnabledEx.

You must call ValidateLicense prior to calling IsFeatureEnabled. C++: VARIANT_BOOL

IsFeatureEnabled (int feature)

C#: bool IsFeatureEnabled (int feature)

Parameters

feature - id of feature to be checked.

Return

Boolean indicating if the featured is enabled.

IsFeatureEnabledEx

Returns whether the specified feature is enabled in this license key. You must call `ValidateLicense` prior to calling `IsFeatureEnabled`.

C++: `VARIANT_BOOL IsFeatureEnabled (int featureSet, int feature)`

C#: `bool IsFeatureEnabledEx (int featureSet, int feature)`

Parameters

featureSet - id of the feature set. QLM supports four feature sets (0 to 3).

feature - id of feature to be checked.

Return

Boolean indicating if the featured is enabled.

IsValid

Returns whether the current license key is a valid key. A valid license key is a key that was decoded properly and is either permanent or evaluation. You must call `ValidateLicense` prior to calling `IsValid`.

C++: `VARIANT_BOOL IsValid`

C#: `bool IsValid()`

Return

Boolean indicating if the license key is a valid key.

LicenseType

Returns the license type of the key. See `ELicenseType` for possible values.

C++: `ELicenseType LicenseType`

C#: `ELicenseType LicenseType ()`

Return

License type

LimitTerminalServerInstances

This attribute limits the number of instances of your application when running on a Terminal Server. When set to true, the QLM ValidateLicense API will fail if the number of running instances is greater than the one defined in the license key.

An instance is defined as an instance of your application with a unique user id and a unique session id. For example, if a user initiates a remote desktop session to a system, and launches 5 instances of your application, the 5 instances will count as one because the same user has launched the 5 instances in the same session.

If the same user then initiates another remote desktop session and launches 3 instances of your application, the total number of instances will be 2.

This feature is only available with QLM Professional and QLM Enterprise.

You can control the number of allowed instances on a terminal server session by setting the Floating Seats property when creating an Activation Key.

NumberOfLicenses

Returns the number of multiple activations enabled for the license key.

C++: int NumberOfLicenses

C#: int NumberOfLicenses;

Return

Number Of LNumber Of Licenses

MajorVersion

Returns the major version associated to the license key. You must call `ValidateLicense` prior to calling this function.

C++: `int MajorVersion`

C#: `int MajorVersion`

Return

Major version of the product.

MinorVersion

Returns the minor version associated to the license key. You must call ValidateLicense prior to calling this function.

C++: int MinorVersion

C#: int MinorVersion

Return

Minor version of the product.

LoadSettings

Loads the settings XML file generated by the Protect Your Application wizard and initializes the QlmLicense object with these settings.

In case of error loading the file, the function will throw an exception.

Note: The settings file format was modified in QLM v9+. Settings files generated by QLM v8 must be first converted to the new format by executing the Protect Your Application wizard in QLM v10.

C#: void LoadSettings (string settingsFile)

Parameters

settingsFile - Xml file generated by the Protect Your Application wizard.

ParseResults

Parses the result of a License Server call. All License Server calls return an XML fragment describing the results of the call. This function parses the results and returns an `ILicenseInfo` interface describing the results.

C#: `public bool ParseResults(string results, ref ILicenseInfo licenseInfo, out string message)`

Parameters

results - value return from any License Server call

licenseInfo - object containing the result of the parse.

message - error messages if an error occurred while parsing the data

PrivateKey

QLM version 5 implements asymmetric encryption to encrypt the license key. Asymmetric encryption is safer because one key encrypts the license, the private key, and another key, the public key, decrypts that information. Therefore, you only need to include the public key in your source code.

This function sets the private key associated with your product. The private key should be set before you create a license, typically right after the call to DefineProduct. If you are creating a license key with a QLM engine version prior to version 5, you do not need to set the private key. It is highly recommended that you do not set the private key in your code.

The private key of your product can be found on the DefineProduct screen under the Keys tab in the QLM Console.

C++: `_bstr_t privateKey`

C#: `string PrivateKey`

ProductID

Returns the product ID associated to the license key. You must call ValidateLicense prior to calling this function.

C++: ibt ProductID

C#: int ProductID ()

Return

Product ID associated to the license key.

ProxyUser

Get or set the name of the user account to use when connecting via a proxy server.

You must also set the following properties: UseProxy, ProxyPassword and ProxyDomain.

ProxyPassword

Get or set the password of the user account to use when connecting via a proxy server. You must also set the following properties: UseProxy, ProxyUser and ProxyDomain.

ProxyDomain

Get or set the domain of the user account to use when connecting via a proxy server.
You must also set the following properties: UseProxy, ProxyUser and ProxyPassword.

PublicKey

QLM version 5 implements asymmetric encryption to encrypt the license key. Asymmetric encryption is safer because one key encrypts the license, the private key, and another key, the public key, decrypts that information. Therefore, you only need to include the public key in your source code.

This function sets the public key associated with your product. The public key should be set before you validate a license, typically right after the call to `DefineProduct`. If you are validating a license key with a QLM engine version prior to version 5, you do not need to set the public key.

The public key of your product can be found on the `DefineProduct` screen under the `Keys` tab in the QLM Console.

C++: `_bstr_t publicKey`

C#: `string PublicKey`

ReadCookie

Reads data stored in a cookie by the StoreCookie API.

Use the FavorMachineLevelLicenseKey to control which key takes precedence if the keys are stored at the user and machine level.

C#: bool ReadCookie(string cookieName, int index, out string data)

Parameters

cookieName- the name of the cookie. index - the index of the element to retrieve. data - the returned data

ReadFloatingLicenseLocation

Reads the license keys stored on the end user system with the StoreFloatingLicenseLocation API.

C#: bool ReadFloatingLicenseLocation (ref string floatingLicenseDbPath)

Parameters

floatingLicenseDbPath- returns the location of the floating license database.

The function returns true if we were able to retrieve the location.

ReadKeys

Reads the license keys stored on the end user system with the StoreKeys API. To store keys, use the StoreKeys API. To clear the stored keys, use the DeleteKeys API.

Use the FavorMachineLevelLicenseKey to control which key takes precedence if the keys are stored at the user and machine level.

C#: void ReadKeys (ref string activationKey, ref string computerKey)

Parameters

activationKey- the stored activation key. computerKey- the stored computer bound key.

StoreCookie

Writes data into a QLM cookie. To read keys, use the ReadKeys API. To clear the stored keys, use the ReadCookie API.

C#: bool StoreCookie(string data, string cookieName, int index, out bool userLevelResult, out bool machineLevelResult, out string errorMessage)

Parameters

- data - the data to write.
- cookieName - a unique name for the cookie.
- index - the index of the element to write. You can write multiple entries into a given cookie.
- userLevelResult- returned boolean value indicating whether the operation was successful at the user level.
- machineLevelResult- returned boolean value indicating whether the operation was successful at the machine level.
- errorMessage- returned error message containing details about the failure, if any.

StoreCookie returns true if either the user level or the machine level operation is successful.

Description

QLM stores its data in two locations: one location at the user level and another location at the machine level. The StoreKeysOptions property controls where data is stored.

Additionally, the StoreKeysLocation property controls whether data is stored on the file system or in the registry.

Data on the file system

If you are running XP, the folders are:

C:\Documents and Settings\<your account name>\Application Data\IsolatedStorage*

C:\Documents and Settings\All Users\Application Data\IsolatedStorage

On Windows 7 or higher, the folders are:

C:\ProgramData\IsolatedStorage

C:\Users\tom\AppData\Local\IsolatedStorage

Example on Windows 7 or higher:

C:\ProgramData\IsolatedStorage\1zy03lmk.jql\epxur3qn.na0\StrongName.gziza0ait44cgjtqq2fgdpi3yp0i
dvio\AssemFiles

Under these folders, a file whose name is the GUID associated to your product (GUID in Define Products page) is created and contains the license keys.

Data in the registry

When QlmLicense.StoreKeysLocation is set to EStoreKeysTo.ERegistry, QLM tries to store the keys in 2 registry hives, one hive at the user level and one hive at the machine level.

On a 32 bit OS, QLM will write license information to:

- HKEY_CURRENT_USER\Software\Classes\CLSID\{GUID}
- HKEY_LOCAL_MACHINE\Software\Classes\CLSID\{GUID}

On a 64 bit OS, QLM will write license information to:

- HKEY_CURRENT_USER\Software\Classes\CLSID\{GUID}
- HKEY_LOCAL_MACHINE\Software\Classes\CLSID\{GUID}
- HKEY_CURRENT_USER\Software\Wow6432Node\Classes\CLSID\{GUID}

- HKEY_LOCAL_MACHINE\Software\Wow6432Node\Classes\CLSID\[GUID]

StoreFloatingLicenseLocation

Stores the location of the floating license database and returns an error message in case of failure. To read the location, use the ReadFloatingLicenseLocation API. To clear the stored information, use the DeleteKeys API.

C#: StoreFloatingLicenseLocation(string floatingLicenseDbPath, out bool userLevelResult, out bool machineLevelResult, out string errorMessage)

Parameters

- floatingLicenseDbPath- full path (in UNC format) to the floating license database.
- userLevelResult- returned boolean value indicating whether the operation was successful at the user level.
- machineLevelResult- returned boolean value indicating whether the operation was successful at the machine level.
- errorMessage- returned error message containing details about the failure, if any.

StoreFloatingLicenseLocation returns true if either the user level or the machine level operation is successful.

Description

QLM stores its data in two locations: one location at the user level and another location at the machine level. The StoreKeysOptions property controls where data is stored.

Additionally, the StoreKeysLocation property controls whether data is stored on the file system or in the registry.

Data on the file system

If you are running XP, the folders are:

C:\Documents and Settings\<your account name>\Application Data\IsolatedStorage*

C:\Documents and Settings\All Users\Application Data\IsolatedStorage

On Windows 7 or higher, the folders are:

C:\ProgramData\IsolatedStorage

C:\Users\tom\AppData\Local\IsolatedStorage

Example on Windows 7 or higher:

C:\ProgramData\IsolatedStorage\1zy03lmk.jql\epxur3qn.na0\StrongName.gziza0ait44cgjtqq2fgdpi3yp0idvio\AssemFiles

Under these folders, a file whose name is the GUID associated to your product (GUID in Define Products page) is created and contains the license keys.

Data in the registry

When QlmLicense.StoreKeysLocation is set to EStoreKeysTo.ERegistry, QLM tries to store the keys in 2 registry hives, one hive at the user level and one hive at the machine level.

On a 32 bit OS, QLM will write license information to:

- HKEY_CURRENT_USER\Software\Classes\CLSID\{GUID}
- HKEY_LOCAL_MACHINE\Software\Classes\CLSID\{GUID}

On a 64 bit OS, QLM will write license information to:

- HKEY_CURRENT_USER\Software\Classes\CLSID\{GUID}
- HKEY_LOCAL_MACHINE\Software\Classes\CLSID\{GUID}
- HKEY_CURRENT_USER\Software\Wow6432Node\Classes\CLSID\{GUID}

- HKEY_LOCAL_MACHINE\Software\Wow6432Node\Classes\CLSID\[GUID]

StoreKeys

Stores license keys on the end user system. To read keys, use the ReadKeys API. To clear the stored keys, use the DeleteKeys API.

C#: void StoreKeys (string activationKey, string computerKey)

Parameters

- activationKey- the activation key to store
- computerKey- the computer bound key to store.

Description

QLM stores its data in two locations: one location at the user level and another location at the machine level. The StoreKeysOptions property controls where data is stored.

Additionally, the StoreKeysLocation property controls whether data is stored on the file system or in the registry.

Data on the file system

If you are running XP, the folders are:

C:\Documents and Settings\<your account name>\Application Data\IsolatedStorage*

C:\Documents and Settings\All Users\Application Data\IsolatedStorage

On Windows 7 or higher, the folders are:

C:\ProgramData\IsolatedStorage

C:\Users\tom\AppData\Local\IsolatedStorage

Example on Windows 7 or higher:

C:\ProgramData\IsolatedStorage\1zy03lmk.jql\epxur3qn.na0\StrongName.gziza0ait44cgjtqq2fgdpi3yp0i
dvio\AssemFiles

Under these folders, a file whose name is the GUID associated to your product (GUID in Define Products page) is created and contains the license keys.

Data in the registry

When QlmLicense.StoreKeysLocation is set to EStoreKeysTo.ERegistry, QLM tries to stores the keys in 2 registry hives, one hive at the user level and one hive at the machine level.

On a 32 bit OS, QLM will write license information to:

- HKEY_CURRENT_USER\Software\Classes\CLSID\[GUID]
- HKEY_LOCAL_MACHINE\Software\Classes\CLSID\[GUID]

On a 64 bit OS, QLM will write license information to:

- HKEY_CURRENT_USER\Software\Classes\CLSID\[GUID]
- HKEY_LOCAL_MACHINE\Software\Classes\CLSID\[GUID]
- HKEY_CURRENT_USER\Software\Wow6432Node\Classes\CLSID\[GUID]
- HKEY_LOCAL_MACHINE\Software\Wow6432Node\Classes\CLSID\[GUID]

StoreKeysEx

Stores the license keys on the computer and returns an error message in case of failure. To read keys, use the ReadKeys API. To clear the stored keys, use the DeleteKeys API.

C#: StoreKeysEx(string activationKey, string computerKey, out bool userLevelResult, out bool machineLevelResult, out string errorMessage)

Parameters

- activationKey- the activation key to store
- computerKey- the computer bound key to store.
- userLevelResult- returned boolean value indicating whether the operation was successful at the user level.
- machineLevelResult- returned boolean value indicating whether the operation was successful at the machine level.
- errorMessage- returned error message containing details about the failure, if any.

StoreKeysEx returns true if either the user level or the machine level operation is successful.

Description

QLM stores its data in two locations: one location at the user level and another location at the machine level. The StoreKeysOptions property controls where data is stored.

Additionally, the StoreKeysLocation property controls whether data is stored on the file system or in the registry.

Data on the file system

If you are running XP, the folders are:

C:\Documents and Settings\<your account name>\Application Data\IsolatedStorage*

C:\Documents and Settings\All Users\Application Data\IsolatedStorage

On Windows 7 or higher, the folders are:

C:\ProgramData\IsolatedStorage

C:\Users\tom\AppData\Local\IsolatedStorage

Example on Windows 7 or higher:

C:\ProgramData\IsolatedStorage\1zy03lmk.jql\epxur3qn.na0\StrongName.gziza0ait44cgjtqq2fgdpi3yp0idvio\AssemFiles

Under these folders, a file whose name is the GUID associated to your product (GUID in Define Products page) is created and contains the license keys.

Data in the registry

When QlmLicense.StoreKeysLocation is set to EStoreKeysTo.ERegistry, QLM tries to store the keys in 2 registry hives, one hive at the user level and one hive at the machine level.

On a 32 bit OS, QLM will write license information to:

- HKEY_CURRENT_USER\Software\Classes\CLSID\{GUID}
- HKEY_LOCAL_MACHINE\Software\Classes\CLSID\{GUID}

On a 64 bit OS, QLM will write license information to:

- HKEY_CURRENT_USER\Software\Classes\CLSID\{GUID}
- HKEY_LOCAL_MACHINE\Software\Classes\CLSID\{GUID}
- HKEY_CURRENT_USER\Software\Wow6432Node\Classes\CLSID\{GUID}
- HKEY_LOCAL_MACHINE\Software\Wow6432Node\Classes\CLSID\{GUID}

StoreKeysLocation

This property affects the WriteKeys API.

The QLM API includes 2 methods that can store and read back the keys: StoreKeys and ReadKeys. If you use the StoreKeys and ReadKeys API, or if you use the QLM .NET Web Control, the activation key and computer key can be stored either in a file or in the registry on the end user system.

The possible values for this property are:

EFile: stores the license key in a file on the end user system.

ERegistry: stores the license key in the registry on the end user system.

To clear the keys stored on the system, call the DeleteKeys API.

StoreKeysOptions

This property affects the ReadKeys API.

The QLM API includes 2 methods that can store and read back the keys: StoreKeys and ReadKeys. If you use the StoreKeys and ReadKeys API, or if you use the QLM .NET Web Control, the activation key and computer key are stored in a file or in the registry on the end user system.

QLM tries to store the keys in 2 locations: 1 location at the user level and 1 location at the machine level.

For example, if you are storing keys to a file and you are running XP, the folders would be:

C:\Documents and Settings\\Application Data\IsolatedStorage\
C:\Documents and Settings\All Users\Application Data\IsolatedStorage

On Vista/Windows7/Windows8, it would be one of these folders:

[C:\ProgramData\IsolatedStorage](#)
C:\Users\AppData\Local\IsolatedStorage or
[C:\Users\AppData\Roaming\IsolatedStorage](#)

Example on Windows 7:

[C:\ProgramData\IsolatedStorage\1zy03lmk.jql\epxur3qn.na0\StrongName.gziza0ait44cgjtqq2fgdpi3yp0i
dvio\AssemFiles](#)

Under these folders, look for a file whose name is the GUID associated to your product (GUID in Define Products page).

When you call the ReadKeys API, ReadKeys tries to load the keys at the user level. If no keys are found at the user level, then QLM tries to read the keys at the machine level.

The **StoreKeysOptions** property controls this behavior. The possible values are: EStoreKeysPerUser, EStoreKeysPerMachine, EStoreKeysPerUserAndMachine..

The QlmLicense.FavorMachineLevelLicenseKey property determines which license key to pick up if QLM finds license keys at the user level as well as the machine level.

To clear the keys stored on the system, call the DeleteKeys API.

UseProxy

Get or set the flag to enable using a proxy server when connecting to the QLM License Server. When set to true, you must also set the following properties: ProxyUser, ProxyPassword and ProxyDomain

ValidateFile

Validates that the Quick License Manager DLL is authentic and was not tampered with. In order to prevent hackers from replacing the IsLicense50.dll with their own version, you can validate the authenticity of the DLL by calling the ValidateFile function. The ValidateFile function returns a fingerprint (long number) that is the result of a checksum of the DLL contents combined with your own key. Use the QlmFingerPrint.exe to generate this unique fingerprint and validate in your code that the runtime fingerprint matches the generated one.

If you are using QLM Professional, you do not need to call this function. Instead, set the validateIntegrity argument to true when constructing the QlmLicense object.

C++: long ValidateFile (BSTR LicenseDLL, BSTR Key);

C#: ulong ValidateFile(string LicenseDLL, BSTR Key)

Parameters

LicenseDLL

: Full path to the License DLL. If this argument is NULL, the currently loaded License DLL is used.

Key

: A unique key of your choice that is used to uniquely encrypt the fingerprint.

Return

FingerPrint- A long umber that uniquely identifies your license DLL.

ValidateLicense

Validates a license key. You must call DefineProduct prior to calling this function.

After calling this function, call GetStatus to get the status of the call.

C++: `_bstr_t ValidateLicense (BSTR LicenseKey);`

C#: `string ValidateLicense (string LicenseKey)`

Parameters

LicenseKey

: License Key to validate

Return

Error message if ValidateLicense fails to validate or if the license is an evaluation license.

ValidateLicenseEx

Validates a computer bound license key. You can call this function for any type of license key. If the license key is not computer bound, set the ComputerID to an empty string. You must call DefineProduct prior to calling this function.

After calling this function, call GetStatus to get the status of the call.

C++: `_bstr_t ValidateLicenseEx (BSTR LicenseKey, BSTR ComputerID);`

C#: `string ValidateLicenseEx (string LicenseKey, string ComputerID)`

Parameters

LicenseKey

: License Key to validate

ComputerID

: A string identifying the computer.

Return

Error message if ValidateLicenseEx fails to validate or if the license is an evaluation license.

ValidateSignature

Validates the signature of a digitally signed XML document. This function can be used to validate the signature of the QLM License Wizard settings XML file as well as the Product Properties XML file.

C#: bool ValidateSignature(string xmlValue, string publicKey, out string errorMessage)

Parameters

xmlValue: The digitally signed XML document to validate.

publicKey: The public key to use when validating the signature. The public key is defined in the QLM Management Console / Define Products / Encryption Keys / Encryption Keys used to digitally sign license files.

errorMessage: If the signature is not valid, the errorMessage may contain details about the failure.

Return

True if the signature is valid.

Version

Returns the version of the QLM engine used to create the key. You must call `ValidateLicense` prior to calling this function.

C++: `_bstr_t` Version

C#: `string` Version

Return

Version of QLM Engine used to create the license key.

WSCredentials

By default, QLM uses anonymous authentication to connect to the QLM License Server.

To connect to the QLM License Server via Windows Authentication, you must set the credentials of the user that will connect to the License Server.

Note that to configure the QLM console to connect to your License Server via Windows Authentication, you must set the Authentication method in the QLM Console / Sites property sheet.

Example:

```
QlmLicense license = new QlmLicense ();  
WSCredentials ws = new WSCredentials ("user", "pwd", "domain");  
license.WSCredentials = ws;
```

GetFirstMACAddress

Gets the MAC address of the first network card found on the system. Note that some computer systems have several network cards. This function returns the MAC address of the first card only. To get a list of all MAC addresses on a system, use the GetMACAddresses method.

C++:

```
_bstr_t GetFirstMACAddress ();
```

C#:

```
string GetFirstMACAddress ();
```

Parameters

None.

Return

Returns the MAC address of the first network card found on the system.

GetMACAddresses

Gets the MAC addresses of all network cards found on the system.

C++:

```
_bstr_t GetMACAddresses ();
```

C#:

```
string GetMACAddresses ();
```

Parameters

None.

Return

Returns a semi colon separated list of MAC addresses.

GetVolumeSerialNumber

Gets the serial number of the specified volume.

C++:

```
_bstr_t GetVolumeSerialNumber (_bstr_t volume);
```

C#:

```
string GetVolumeSerialNumber (string volume);
```

Parameters

volume: volume driver letter such as C:.

Return

Returns the serial number of the specified volume.

RunningOnHyperV

Determines if the current process is running on a Microsoft Hyper-V virtual machine.

C++:

BOOL RunningOnHyperV ();

C#:

bool RunningOnHyperV ();

Parameters

None.

Return

Returns true if the current process is running on a Microsoft Hyper-V virtual machine.

RunningOnVirtualBox

Determines if the current process is running on a Virtual Box virtual machine.

C++:

BOOL RunningOnVirtualBox ();

C#:

bool RunningOnVirtualBox ();

Parameters

None.

Return

Returns true if the current process is running on a Virtual Box virtual machine.

RunningOnVM

Determines if the current process is running on a virtual machine such as Microsoft Hyper-V or VMWare.

C++: BOOL RunningOnVM ();

C#: bool RunningOnVM ();

Parameters

None.

Return

Returns true if the current process is running on a Virtual Machine such as Microsoft Hyper-V or VMWare.

RunningOnVMWare

Determines if the current process is running on a VMWare virtual machine.

C++:

BOOL RunningOnVMWare ();

C#:

bool RunningOnVMWare ();

Parameters

None.

Return

Returns true if the current process is running on a VMWare virtual machine.

HTTP Methods

The HTTP Methods are methods that can be invoked via a URL. These methods are typically invoked from your eCommerce provider during the purchase process.

Note that all other methods exposed by the QLM License Server cannot be called directly via SOAP. In order to communicate with the QLM License Server, you need to use the QLM .NET API methods that are exposed via the QLMLicenseLib.dll.

ActivateKey

Activates a key over the internet.

To invoke this method via a URL:

`http://yourserver/yourvirtualdirectory/qlmservice.asmx/ActivateKey?is_avkey=ABCD-EFGH-IJKL&is_pcid=123456&is_productid=<productID>&is_majorversion=<majorVersion>&is_minorversion=<minorVersion>&is_vendor=fastspring`

where

- `is_avkey` = Activation Key
- `is_pcid` = Unique identifier of the computer
- `is_computer_name` = Name of the computer (optional)
- `is_vendor` = One of the supported vendors
- `is_productid` = your product id as defined in QLM
- `is_majorversion` = your product's major version as defined in QLM
- `is_minorversion` = your product's minor version as defined in QLM
- `is_email` = Email address of the customer associated to this key (optional)
- `is_userdata1` = User data to associate to the key (optional)
- `is_affiliateid` = Affiliate to associate to the key (optional)
- `is_qlmversion` = 5.0.00 or earlier versions (optional)
- `is_user` = username defined for the selected eCommerce provider (optional)
- `is_pwd` = password defined for the selected eCommerce provider (optional)

AnalyticsAddInstallHttp

Registers an installation with the server. You should call this function once when your application is installed. You should store the returned installID in your application's settings and reuse it on subsequent calls to the QlmAnalytics API. Note that to call this function, you must: Have a QLM Enterprise License
To invoke this method via a URL:

`http://yourserver/yourvirtualdirectory/qlmservice.asmx/AnalyticsAddInstallHttp?is_swversion=1.0&is_osversion=Windows 7&...`

;

where

- `is_swversion`= Version of your software
- `is_osversion`= Version of the operating system
- `is_computer_name` = Name of the computer
- `is_computerID`= Unique identifier of the computer
- `is_avkey` = activation key on the system
- `is_pckey`= computer key associated to the system
- `is_trial`= flag indicating if the license is a trial
- `is_productname`= name of your product
- `is_majorversion` = major version of your product
- `is_minorversion`= minor version of your product
- `is_customdata1`= your own custom data
- `is_customdata2`= your own custom data
- `is_customdata3`= your own custom data

AnalyticsRemoveInstallHttp

Unregisters an application with the server. You should call this function when the user uninstalls your application. Note that to call this function, you must: Have a QLM Enterprise License

To invoke this method via a URL:

http://yourserver/yourvirtualdirectory/qlmservice.asmx/AnalyticsRemoveInstallHttp?is_installid=35BC5B3C-9102-46D5-BF75-29137A1F97E6

where

- is_installid = ID of the installation returned by the call to AnalyticsAddInstallHttp

AnalyticsUpdateInstallHttp

Updates information of a registered installation on the server. Note that to call this function, you must:
Have a QLM Enterprise License

To invoke this method via a URL:

http://yourserver/yourvirtualdirectory/qlmservice.asmx/AnalyticsUpdateInstallHttp?is_installid=35BC5B3C-9102-46D5-BF75-29137A1F97E6&is_osversion=Windows 7&...

;

where

- is_installid = ID of the installation returned by the call to AnalyticsAddInstallHttp
- is_swversion= Version of your software
- is_osversion= Version of the operating system
- is_computer_name = Name of the computer
- is_computerID= Unique identifier of the computer
- is_avkey = activation key on the system
- is_pckey= computer key associated to the system
- is_trial= flag indicating if the license is a trial
- is_productname= name of your product
- is_majorversion = major version of your product
- is_minorversion= minor version of your product
- is_customdata1= your own custom data
- is_customdata2= your own custom data
- is_customdata3= your own custom data

AnalyticUpdateLastAccessedDateHttp

Updates the Last Accessed Date associated to a registered installation on the server. Note that to call this function, you must: Have a QLM Enterprise License

To invoke this method via a URL:

`http://yourserver/yourvirtualdirectory/qlmservice.asmx/AnalyticUpdateLastAccessedDateHttp?is_installid=35BC5B3C-9102-46D5-BF75-29137A1F97E6`

where

- `is_installid` = ID of the installation returned by the call to `AnalyticsAddInstallHttp`

EnableMaintenancePlan

Enables the maintenance plan for a given activation key.

To invoke this method via a URL:

`http://yourserver/yourvirtualdirectory/qlmservice.asmx/EnableMaintenancePlan?is_avkey=<activationKey>&is_maintplan=1&is_vendor=<vendor>&is_maintduration=<duration>`
where

- `is_vendor` = One of the supported vendors
- `is_avkey` = Activation Key
- `is_maintplan` = Flag to determine if the maintenance plan should be extended. If this argument is specified, the maintenance plan is extended.
- `is_maintdate` = Date at which the new maintenance plan should expire
- `is_maintduration` = number of days by which to extend the maintenance plan as of today.
- `is_user` = username defined in Manage Keys / 3rd Party Extensions (optional)
- `is_pwd` = password defined in Manage Keys / 3rd Party Extensions (optional)

If neither `is_maintdate` nor `is_maintduration` are specified, the maintenance plan is extended based on the `maintenancePlanPeriodInDays` settings in the QLM License Server config file (`web.config`). The default value of `maintenancePlanPeriodInDays` is 365 days.

If `is_maintdate` and `is_maintduration` are both specified, `is_maintdate` takes precedence.

The format of the date in `is_maintdate` is based on the `dateFormat` settings in the License Server config file (`web.config`). The default format is: YYYY-MM-dd.

When invoking this method from an eCommerce provider, you can customize the url arguments for `is_avkey` and `is_maintplan`. Customization of these arguments requires subclassing of the eCommerce provider classes. For more details, contact us.

GetActivationKey

Creates an activation key over the internet.

Note that to call this function, you must update the web.config on the web server as follows:

```
<setting name="enableGetActivationKey" serializeAs="String">  
<value>True</value>  
</setting>
```

To invoke this method via a URL:

http://yourserver/yourvirtualdirectory/qlmservice.aspx/GetActivationKey?is_productid=<productID>&is_majorversion=<majorVersion>&is_minorversion=<minorVersion>&is_vendor=fastspring
where

- is_vendor = One of the supported vendors
- is_productid = your product id as defined in QLM
- is_majorversion = your product's major version as defined in QLM
- is_minorversion = your product's minor version as defined in QLM
- is_qlmversion = 5.0.00 or earlier versions
- is_quantity = the number of licenses to create. Works in conjunction with is_usemultipleactivationskey. Note that when calling this API from one of the supported eCommerce Providers, you do not need to set is_quantity as it is automatically passed from the eCommerce provider to QLM.

Optional Arguments:

- is_features: semi comma separated list of feature sets and their corresponding values. Example: is_features=0:1;1:2;2:3;3:6.
- is_usemultipleactivationskey: when set to true or not set, if multiple licenses are ordered in the same request, the system returns one activation key for all licenses. When set to false, the system returns one activation key for each ordered license. The default is true. Example: &is_usemultipleactivationskey=true
- is_additionalactivations: add a fixed number of activations to the generated license. Example: is_additionalactivations=2
- is_numberofactivationsperkey: for each generated key, set the number of activations allowed. Example: is_numberofactivationsperkey=3. This argument is only effective when is_usemultipleactivationskey is false.
- is_user: username as defined in the eCommerce Providers section in QLM (Manage Keys / Tools / eCommerce Providers). Example: &is_user=tom
- is_pwd: password as defined in the eCommerce Providers section in QLM (Manage Keys / Tools / eCommerce Providers). Example: &is_pwd=pwd
- is_maintenance_plan: Set this value to 1 to enable the maintenance plan for this license. Example: &is_maintenance_plan=1
- is_maintduration: Set this value to the duration of the maintenance plan. By default, a maintenance plan is 365 days. The default value can be changed in the web.config file of the License Server. Example: &is_maintduration=180
-
- is_additionalactivations: By default, the number of activations enabled for each generated license key is based on the quantity of licenses purchased. For example, if a customer purchases 3 licenses, they will receive an activation key with 3 activations (is_usemultipleactivationskey must be true). To provide additional activations, set this argument to the number of additional activations that you require. Example: &is_additionalactivations=3

GetActivationKeyWithExpiryDate

Creates an activation key with an expiry date over the internet.

Note that to call this function, you must update the web.config on the web server as follows:

```
<setting name="enableGetActivationKey" serializeAs="String">
<value>True</value>
</setting>
```

To invoke this method via a URL:

http://yourserver/yourvirtualdirectory/qlmservice.asmx/GetActivationKeyWithExpiryDate?is_productid=
<productID>&is_majorversion=<majorVersion>&is_minorversion=<minorVersion>&is_vendor=
fastspring&is_expduration=<duration>&is_expdate=<date>
where

- is_vendor = One of the supported vendors
- is_productid = your product id as defined in QLM
- is_majorversion = your product's major version as defined in QLM
- is_minorversion = your product's minor version as defined in QLM
- is_qlmversion = 5.0.00 or earlier versions
- is_expduration = duration of evaluation version in days.
- is_expdate = date at which the license will expire. The format of the date in *is_expdate* is based on the *dateFormat* settings in the License Server config file (web.config). The default format is: YYYY-MM-dd
- is_quantity = the number of licenses to create. Works in conjunction with *is_usemultipleactivationskey*. Note that when calling this API from one of the supported eCommerce Providers, you do not need to set *is_quantity* as it is automatically passed from the eCommerce provider to QLM.

When calling GetActivationWithExpiryDate, an expiry date can be set by using the *is_expduration* or *is_expdate* argument. For example, to create a license key that expires 31 days after purchase, you have 2 options:

- Use the *is_expdate* argument such as: &is_expdate="2015-03-01". The generated license will expire on the specified date.
- Use the *is_expduration* argument such as: &is_expduration=31.

The *useDurationToSetExpiryDate* setting in the QLM License Server "Server Properties" determines the behavior of the *is_expduration* argument. When *useDurationToSetExpiryDate* is true, an expiry date is computed by the License Server based on the order date plus the *is_expduration* period. For example, if a user purchased a 30 days subscription of your product on January 1st and *is_expduration* is set to 30, the license will be set to expire on January 31st.

When *useDurationToSetExpiryDate* is set to false, the generated license key does not have a specific expiry date but rather a duration based license key. This means that the license expiry date is determined the first time the application is executed. For example, if a user purchased a 30 day subscription of your product on January 1st, but then runs your product for the first time on January 15, the 30 day license will only start on January 15.

Optional Arguments:

- *is_features*: semi comma separated list of feature sets and their corresponding values. Example: *is_features=0:3;1:1*. This means that in feature set 0, features 1 + 2 are enabled and in feature set 1, feature 1 is enabled.
- *is_usemultipleactivationskey*: when set to true or not set, if multiple licenses are ordered in the same request, the system returns one activation key for all licenses. When set to false, the system returns one activation key for each ordered license. The default is true. Example:
&is_usemultipleactivationskey=true
- *is_additionalactivations*: add a fixed number of activations to the generated license. Example:

is_additionalactivations=2

- is_numberofactivationsperkey: for each generated key, set the number of activations allowed. Example: is_numberofactivationsperkey=3. This argument is only effective when is_usemultipleactivationskey is false.
- is_user: username as defined in the eCommerce Providers section in QLM (Manage Keys / Tools / eCommerce Providers). Example: &is_user=tom
- is_pwd: password as defined in the eCommerce Providers section in QLM (Manage Keys / Tools / eCommerce Providers). Example: &is_pwd=pwd
- is_maintenance_plan: Set this value to 1 to enable the maintenance plan for this license. Example: &is_maintenance_plan=1
- is_maintduration: Set this value to the duration of the maintenance plan. By default, a maintenance plan is 365 days. The default value can be changed in the web.config file of the License Server. Example: &is_maintduration=180
- is_additionalactivations: By default, the number of activations enabled for each generated license key is based on the quantity of licenses purchased. For example, if a customer purchases 3 licenses, they will receive an activation key with 3 activations (is_usemultipleactivationskey must be true). To provide additional activations, set this argument to the number of additional activations that you require. Example: &is_additionalactivations=3
- is_licensemodel: the license model can be one of: permanent | trial | subscription. Example: &is_licensemodel=subscription
- is_computertype: specifies the type of the client computer. Possible values are: 0 | 1 | 2 . 0 sets the computer type to 'none'. 1 sets the computer type to 'PC'. 2 sets the computer type to 'VM'.

ReleaseLicenseHttp

Releases a license key over Http. This API releases a license key so that it can be activated on another computer.

To invoke this method via a URL:

`http://yourserver/yourvirtualdirectory/qlmservice.asmx/ReleaseLicenseHttp?is_avkey=<activationKey>
&is_pcid=<computer ID>&is_vendor=<eCommerce provider>`

where

- `is_avkey` = Activation key to deactivate.
- `is_pcid` = Computer identifier of the system being deactivated. This argument is required if the license is a multiple activations key.
- `is_vendor` = name of the eCommerce provider
- `is_user`: username as defined in the eCommerce Providers section in QLM (Manage Keys / Tools / eCommerce Providers). Example: `&is_user=tom`
- `is_pwd`: password as defined in the eCommerce Providers section in QLM (Manage Keys / Tools / eCommerce Providers). Example: `&is_pwd=pwd`

RenewMaintenancePlan

Connects to the License Server and renews a maintenance plan for a given activation key. By default, the maintenance plan is renewed with the period specified in the QLM Management Console / Manage Keys / Sites / Server Properties / maintenancePlanPeriodInDays. The following arguments are required:

- is_avkey - The Activation Key to renew
- is_vendor - One of the supported eCommerce providers

The following arguments are optional:

- is_maintduration - duration in days of the maintenance plan
- is_user = username as defined in the eCommerce Providers section in QLM (Manage Keys / Tools / eCommerce Providers) - This argument is optional
- is_pwd = password as defined in the eCommerce Providers section in QLM (Manage Keys / Tools / eCommerce Providers) - *This argument is optional*

RenewSubscriptionHttp

Connects to the License Server and renews a subscription. When a subscription is renewed, each activated license is automatically reactivated on the server and a new computer bound key is generated with a new expiry date. When customers reactivate their license, they receive the new computer bound key with the new expiry date, thus extending their subscription period.

To invoke this method via a URL:

`http://yourserver/yourvirtualdirectory/qlmservice.asmx/RenewSubscriptionHttp?is_avkey=<activationKey>&is_vendor=<xyz>&is_expdate=<date>&is_user=<user>&is_pwd=<pwd>`
where

- `is_avkey` = Activation Key that is being renewed. This argument is required.
- `is_vendor` = One of the supported eCommerce providers. This argument is required.
- `is_expdate` = date at which the license will expire. You must specify either `is_expdate` or `is_expduration`.
- `is_expduration` = duration in days after which the license will expire. You must specify either `is_expdate` or `is_expduration`.
- `is_user` = username as defined in the eCommerce Providers section in QLM (Manage Keys / Tools / eCommerce Providers) - This argument is optional
- `is_pwd` = password as defined in the eCommerce Providers section in QLM (Manage Keys / Tools / eCommerce Providers) - *This argument is optional*

RevokeLicenseHttp

Revokes a license key over Http. This API revokes a license key so that it can no longer be activated.

To invoke this method via a URL:

`http://yourserver/yourvirtualdirectory/qlmservice.asmx/RevokeLicenseHttp?is_avkey=<activationKey>
&is_vendor=<eCommerce provider>`

where

- is_avkey= Activation key to revoke.
- is_vendor = name of the eCommerce provider
- is_user: username as defined in the eCommerce Providers section in QLM (Manage Keys / Tools / eCommerce Providers). Example: &is_user=tom
- is_pwd: password as defined in the eCommerce Providers section in QLM (Manage Keys / Tools / eCommerce Providers). Example: &is_pwd=pwd

SubscribeToMailListHttp

Subscribes or unsubscribes a user from the mail list.

To invoke this method via a URL:

http://server/qlm/qlmservice.asmx/SubscribeToMailListHttp?is_email=user@cie.com&is_include=1

;

where

- is_email = email of customer
- is_include = 1 to subscribe, 0 to unsubscribe

UpdateUserInfo

Updates user information in the QLM License Server.

Note that to call this function, you must update the web.config on the web server as follows:

```
<setting name="enableGetActivationKey" serializeAs="String">  
<value>True</value>  
</setting>
```

To invoke this method via a URL:

`http://yourserver/yourvirtualdirectory/qlmservice.asmx/UpgradeLicense?is_vendor=digibuy&is_user=
<user_name>&is_pwd=<user_pwd>`
where

- is_vendor = One of the supported vendors
- is_user = username as defined in the eCommerce Providers section in QLM (Manage Keys / Tools / eCommerce Providers)
- is_pwd = password as defined in the eCommerce Providers section in QLM (Manage Keys / Tools / eCommerce Providers)

The following user related data must be sent as part of the POST data.

- FullName
- Email
- Phone
- Company
- Address1
- Address2
- City
- State
- Zip
- Country
- IP Address
- Notes
- Affiliate Name

The actual field names are e-commerce provider dependent. The supported e-commerce providers are listed on our web site.

Additionally, if the is_avkey argument is added to the URL and specifies an Activation Key that is already published to the QLM License Server, this function will associate the provided Activation Key to the provided user.

UpgradeLicense

Upgrades a license by issuing a new license key and replacing the old one. You can upgrade the following data associated to a license:

- Features associated to a license
- Expiry date of the license
- Duration of the license
- Major and Minor version of the product
- The version of the QLM Engine used to generate the license key

Note that to call this function, you must set the Server Property **enableUpgradeLicense** to true.

To invoke this method via a URL:

`http://yourserver/yourvirtualdirectory/qlmservice.asmx/UpgradeLicense?is_productid=<productID>&is_majorversion=<majorVersion>&is_minorversion=<minorVersion>&is_vendor=digibuy&is_features=<features>&is_expduration=<duration>&is_expdate=<date>`

where

- `is_avkey` = Activation Key that is being upgraded. This argument is required.
- `is_vendor` = One of the supported vendors
- `is_productid` = your product id as defined in QLM
- `is_majorversion` = your product's major version as defined in QLM
- `is_minorversion` = your product's minor version as defined in QLM
- `is_qlmversion` = 5.0.00 or earlier versions
- `is_features` = semi comma separated list of feature sets and their corresponding values. Example: `is_features=0:3;1:1`. This means that in feature set 0, features 1 + 2 are enabled and in feature set 1, feature 1 is enabled.
- `is_expduration` = duration of evaluation version
- `is_expdate` = date at which the license will expire
- `is_user` = username as defined in the eCommerce Providers section in QLM (Manage Keys / Tools / eCommerce Providers)
- `is_pwd` = password as defined in the eCommerce Providers section in QLM (Manage Keys / Tools / eCommerce Providers)

ValidateLicenseHttp

Validates a license key over Http. This API validates and activates a license.

The first time you call ValidateLicenseHttp, you provide the activation key (is_avkey) and the computer identifier (is_pcid). The server does the following:

- Validates the license
- Verifies if the license has not been previously activated
- Activates the license
- Returns the computer key and the set of features that are enabled.

On subsequent calls to ValidateLicenseHttp, in addition to the previous arguments, you should set the computer key argument using the value returned from the first call. In this instance, the server does the following:

- Validates the license
- Verifies if the license has not been revoked
- Returns the status of the license and the set of features that are enabled.

To invoke this method via a URL:

`http://yourserver/yourvirtualdirectory/qlmservice.asmx/ValidateLicenseHttp?is_avkey=<activationKey> &is_pckey=<computer key> &is_pcid=<computer ID> &is_computer_name=<computer name> &is_qlmversion=<QLM Engine version> &is_email=<email of the customer associated to the key> &is_userdata1=<user data to associate to the key> &is_affiliateid=<affiliate to associate to the key> &is_activate=<true | false> &s_writebom=<true | false>`

where

- is_avkey= Activation key to validate. If the key has never been activated, the key will be activated and a computer key will be returned.
- is_pckey = If the key has been previously activated, the ValidateLicense method returns a computer key. This computer key should then be used in subsequent calls to ValidateLicense in the is_pckey argument.
- is_pcid = If the license has never been validated, you need to specify a computer identifier so that the returned computer key can be bound to this specific computer. A computer ID can be the name of the computer or any other unique identifier of your choice.
- is_computer_name= This argument is not required. It is used to easily identify a computer, in case the computer ID is a serial number such as the hard disk serial number.
- is_qlmversion = Version of the QLM engine. IT can be 5.0.00 or earlier versions.
- is_email = Email address of the customer associated to the license key. This argument is optional.
- is_userdata1 = User data to associate to the license key. This argument is optional.
- is_affiliateid = Affiliate to associate to the license key .This argument is optional.
- is_activate = By default, ValidateLicenseHttp validates the license and activates it if needed. If you just want to validate the license and return information about the license, set the is_activate argument to false. If is_activate is not specified, the default value is true.
- is_writebom = Determines whether the byte-order-mark (BOM) is written in the returned xml fragment. If no argument is specified, the default value is taken from the License Server's web.config file "writeBOM" setting. The default value is false.

Optional but recommended arguments:

- is_productid: the ID of the product
- is_majorversion: the major version of the product
- is_minorversion: the minor version of the product

Note that the 3 optional arguments above are required if you use the QLM Maintenance Plan feature.

.NET Control

Following is a list of all the properties that can be set on the QLM .NET Control.

Name	Description
QlmCloseButtonVisible	Show or hide to Close button
QlmComputerID	Set the computer ID to use when activating the license. This property should be typically set programmatically at runtime.
QlmEncryptionKey	Set the encryption key. This property is only required when using QLM engine version 4.0 and earlier.
QlmEvaluationHaveKeyRadioButtonText	Only applies if the QlmEvaluationVisible property is set to true. Set the text in the radio button when the user has a license key.
QlmEvaluationLicenseKey	Only applies if the QlmEvaluationVisible property is set to true. Set the evaluation key to use when the user selects to evaluate the software and does not have a license key.
QlmEvaluationTrialChecked	Only applies if the QlmEvaluationVisible property is set to true. Checks the evaluation option by default.
QlmEvaluationTrialHelpText	Only applies if the QlmEvaluationVisible property is set to true. Set the text to display under the evaluation radio button.
QlmEvaluationTrialRadioButtonText	Only applies if the QlmEvaluationVisible property is set to true. Set the text to display on the evaluation radio button.
QlmEvaluationVisible	Enables the evaluation option. The evaluation option displays two radio buttons. One radio button allows the user to enter a license key and activate the license while the other radio button allows the user to evaluate the software by using an embedded evaluation license key.
QlmFormBackColor	Set the starting Background Color of the form to produce a gradient effect.
QlmFormBackColor2	Set the ending Background Color of the form to produce a gradient effect.
QlmGUID	Set the GUID associated to your product. The

	GUID can be found on the Define Product page in the QLM Console.
QlmHeaderBackColor	Set the Background Color of the header pane.
QlmLicenseStatus	Get the status of the license after it has been validated. This is a read-only property.
QlmLicenseType	Set the license type. The license type can be: ComputerName, UserDefined or Generic.
QlmLogoFont	Set the font to use in the logo text.
QlmLogoImage	Set the image to use for the logo.
QlmLogoText	Set the text to use for the logo.
QlmMajorVersion	Set the Major Version associated to your product. The Major Version can be found on the Define Products page in the QLM Console.
QlmMinorVersion	Set the Minor Version associated to your product. The Minor Version can be found on the Define Products page in the QLM Console.
QlmProductID	Set the Product ID Version associated to your product. The Product ID can be found on the Define Products page in the QLM Console.
QlmProductName	Set the Product Name associated to your product.
QlmProxyButtonVisible	Show or hide the proxy settings button
QlmPublicKey	Set the Public Key associated to your product. The Public Key Version can be found on the Define Products page (Keys tab) in the QLM Console.
QlmStoreKeysLocation	By default, QLM stores the license keys in a hidden file on the end user system. You can also select to store the license keys in the registry by setting this property.
QlmValidateCertificate	The QLM DLLs are digitally signed by a trusted certificate authority. In order to ensure that hackers do not replace the QLM DLLs by dummy ones, QLM can validate that the DLLs are properly signed.

QlmCommunicationEncryptionKey

Set the communicationEncryptionKey to use when connecting to the QLM License Server. The communicationEncryptionKey must match the one defined in the web.config file on the web server.

QlmWebServiceUrl

Set the URL to the QLM License Server. The value of this url is typically:

<http://yourdomain.com/qlm/qlmservice.asmx>

.NET Control

Following is a list of all the properties that can be set on the QLM .NET Control.

Name	Description
QlmCloseButtonVisible	Show or hide to Close button
QlmComputerID	Set the computer ID to use when activating the license. This property should be typically set programmatically at runtime.
QlmEncryptionKey	Set the encryption key. This property is only required when using QLM engine version 4.0 and earlier.
QlmEvaluationHaveKeyRadioButtonText	Only applies if the QlmEvaluationVisible property is set to true. Set the text in the radio button when the user has a license key.
QlmEvaluationLicenseKey	Only applies if the QlmEvaluationVisible property is set to true. Set the evaluation key to use when the user selects to evaluate the software and does not have a license key.
QlmEvaluationTrialChecked	Only applies if the QlmEvaluationVisible property is set to true. Checks the evaluation option by default.
QlmEvaluationTrialHelpText	Only applies if the QlmEvaluationVisible property is set to true. Set the text to display under the evaluation radio button.
QlmEvaluationTrialRadioButtonText	Only applies if the QlmEvaluationVisible property is set to true. Set the text to display on the evaluation radio button.
QlmEvaluationVisible	Enables the evaluation option. The evaluation option displays two radio buttons. One radio button allows the user to enter a license key and activate the license while the other radio button allows the user to evaluate the software by using an embedded evaluation license key.
QlmFormBackColor	Set the starting Background Color of the form to produce a gradient effect.
QlmFormBackColor2	Set the ending Background Color of the form to produce a gradient effect.
QlmGUID	Set the GUID associated to your product. The

	GUID can be found on the Define Product page in the QLM Console.
QlmHeaderBackColor	Set the Background Color of the header pane.
QlmLicenseStatus	Get the status of the license after it has been validated. This is a read-only property.
QlmLicenseType	Set the license type. The license type can be: ComputerName, UserDefined or Generic.
QlmLogoFont	Set the font to use in the logo text.
QlmLogoImage	Set the image to use for the logo.
QlmLogoText	Set the text to use for the logo.
QlmMajorVersion	Set the Major Version associated to your product. The Major Version can be found on the Define Products page in the QLM Console.
QlmMinorVersion	Set the Minor Version associated to your product. The Minor Version can be found on the Define Products page in the QLM Console.
QlmProductID	Set the Product ID Version associated to your product. The Product ID can be found on the Define Products page in the QLM Console.
QlmProductName	Set the Product Name associated to your product.
QlmProxyButtonVisible	Show or hide the proxy settings button
QlmPublicKey	Set the Public Key associated to your product. The Public Key Version can be found on the Define Products page (Keys tab) in the QLM Console.
QlmStoreKeysLocation	By default, QLM stores the license keys in a hidden file on the end user system. You can also select to store the license keys in the registry by setting this property.
QlmValidateCertificate	The QLM DLLs are digitally signed by a trusted certificate authority. In order to ensure that hackers do not replace the QLM DLLs by dummy ones, QLM can validate that the DLLs are properly signed.

QlmCommunicationEncryptionKey

Set the communicationEncryptionKey to use when connecting to the QLM License Server. The communicationEncryptionKey must match the one defined in the web.config file on the web server.

QlmWebServiceUrl

Set the URL to the QLM License Server. The value of this url is typically:

<http://yourdomain.com/qlm/qlmservice.asmx>

QLM Wizard .NET Control Reference

Following is a list of all the properties that can be set on the QLM License Wizard .NET Control.

QLM Wizard - UI Properties

Name	Description
QlmActivationMethodCaption	Caption of the Activation Method page.
QlmActivationMethodCaptionFont	Font of the caption of the Activation Method page.
QlmActivationMethodCaptionForeColor	Foreground color of the caption of the Activation Method page.
QlmActivationMethodTitle	Title of the Activation Method page.
QlmActivationMethodTitleFont	Font of the title of the Activation Method page.
QlmActivationMethodTitleForeColor	Foreground color of the title of the Activation Method page.
QlmActivationMethodTitleImage	Image displayed to the left of the title of the Activation Method page.
QlmBuyNowCaption	Caption of the Buy Now button.
QlmBuyNowUrl	URL associated to the Buy Now button.
QlmCloseFormOnFinish	Automatically close the form when the user clicks the Finish or Cancel button.
QlmDeactivateCaption	Caption of the Deactivate radio button on the Activation Method page.
QlmFontButtons	Font to use on all buttons.
QlmFontLabels	Font to use on all labels.
QlmFontRadioButtons	Font to use on all radio buttons.
QlmFontResult	Font to use in the result panel.
QlmFontText	Font to use on all text fields.
QlmForeColorControls	Foreground colors of all controls such as labels and buttons.
QlmForeColorResult	Foreground colors of the result panel that displays the result of the license validation.
QlmForeColorText	Foreground colors of all text fields.
QlmFormatLicenseKey	Formats the license key by adding dash separators.
QlmFormatLicenseKeyGroupSize	Size of the group of characters in a formatted license. The QlmFormatLicenseKey property must be true for this to take effect.
QlmFormBackColor	Sets the background color of the wizard.
QlmFormBackColor2	Sets the 2nd gradient background color of the wizard.
QlmFormCurvature	Curvature of the corners of the wizard.
QlmGlowColor	Background color of text fields when in focus.
QlmMessageExpired	Sets the text to display if the license key expired.
QlmMessageRemainingDays	Sets the text to display when a trial license is still valid. The message should include a {0} argument to display the remaining days.
QlmMessageTrial	Sets the text to display when a trial license is

QlmOfflineActivationCaption	detected. Caption of the Offline Activation radio button.
QlmOnlineActivationCaption	Caption of the Online Activation radio button.
QlmProductFont	Set the font to use for the Product Name.
QlmProductForecolor	Set the foreground color of the Product Name.
QlmProductImage	Set the image displayed to the left of the product name.
QlmProductTitle	Sets the title of the product.
QlmShowBuyNowButton	Show or hide the Buy Now button.
QlmShowBuyDeactivate	Show or hide the Deactivate radio button.
QlmShowProxyButton	Show or hide the Proxy Settings button.
QlmShowTryButton	Show or hide the Try button.
QlmWizardTitle	Title of the wizard.
QlmWizardTitleIcon	Icon of the wizard.

QLM Wizard - License properties

Name	Description
QlmCommunicationEncryptionKey	Set the communicationEncryptionKey to use when connecting to the QLM License Server. The communicationEncryptionKey must match the one defined in the web.config file on the web server.
QlmEnableMultiByte	Enable multibyte to support system with a ComputerID that contains multibyte characters. By default, QLM sends custom headers with every SOAP request and sends information to the server related to the customer's locale. If this interferes with your own SOAP extension, you can turn off QLM's extension.
QlmEnableSoapExtension	Set the evaluation key to use when the user selects to evaluate the software and does not have a license key.
QlmEvaluationLicenseKey	Sets whether the evaluation information is stored per user or per machine.
QlmEvaluationPerUser	If a license key is stored both at the user level and the machine level, set this property to true to favor the key stored at the machine level.
QlmFavorMachineLevelLicenseKey	Set the GUID associated to your product. The GUID can be found on the Define Product page in the QLM Console.
QlmGUID	Set the license type. The license type can be: ComputerName, UserDefined or Generic.
QlmLicenseType	Set the Major Version associated to your product. The Major Version can be found on the Define Products page in the QLM Console.
QlmMajorVersion	Set the Minor Version associated to your product. The Minor Version can be found on the Define
QlmMinorVersion	

QlmOverrideKeyStoreRegistry	Products page in the QLM Console. Change the default registry key where QLM stores license key information. This is strictly for permanent licenses.
QlmProductID	Set the Product ID Version associated to your product. The Product ID can be found on the Define Products page in the QLM Console.
QlmProductName	Set the Product Name associated to your product.
QlmPublicKey	Set the Public Key associated to your product. The Public Key Version can be found on the Define Products page (Keys tab) in the QLM Console.
QlmStoreKeysLocation	Location where to store license keys. Keys can be stored in a file or in the registry.
QlmStoreKeysOptions	Set whether to store license keys per user, per machine or both.
QlmValidateCertificate	The QLM DLLs are digitally signed by a trusted certificate authority. In order to ensure that hackers do not replace the QLM DLLs by dummy ones, QLM can validate that the DLLs are properly signed.
QlmVersion	Version of the QLM License Engine to use. The recommended value is 5.0.00.
QlmLicenseServerUrl	Set the URL to the QLM License Server. The value of this url is typically: http://yourdomain.com/qlm/qlmservice.asmx

QLM License Wizard .NET Control Reference

Following is a list of all the properties that can be set on the QLM License Wizard .NET Control.

UI Properties

Name	Description
QlmBackColor	Sets the background color of the wizard.
QlmBackColor2	Sets the 2nd gradient background color of the wizard.
QlmIntroductionText97	When the QlmWizardStyle property is set to Wizard97, sets the description text under the title.
QlmLeftPanelImage97	When the QlmWizardStyle property is set to Wizard97, sets the image to use for the left panel.
QlmMainPanelImage	Sets the image to use for the main panel.
QlmMessageExpired	Sets the text to display if the license key expired.
QlmMessageRemainingDays	Sets the text to display when a trial license is still valid. The message should include a {0} argument to display the remaining days.
QlmMessageTrial	Sets the text to display when a trial license is detected.
QlmProceedText97	When the QlmWizardStyle property is set to Wizard97, sets the text to display for proceeding to the next page in the wizard.
QlmProductFont	Set the font to use for the Product Name.
QlmTitleBackColor	Sets the background color of the title.
QlmTitleBackColor2	Sets the 2nd gradient background color of the title.
QlmTitleFirstPage	Sets the title of the first page of the wizard.
QlmTitleForm	Sets the title of the wizard form.
QlmTitleProduct	Sets the title of the product section.
QlmTitleWizard	Sets the title of the wizard section.
QlmWizardStyle	Sets the style of the wizard. The options are: WizardAero or Wizard97.

QLM License object properties

Name	Description
QlmEncryptionKey	Set the encryption key. This property is only required when using QLM engine version 4.0 and earlier.
QlmEvaluationLicenseKey	Set the evaluation key to use when the user selects to evaluate the software and does not have a license key.
QlmEvaluationPerUser	Sets whether the evaluation information is stored per user or per machine.
QlmFavorMachineLevelLicenseKey	If a license key is stored both at the user level and the machine level, set this property to true to favor the key stored at the machine level.

QlmFormatLicenseKey	Formats the license key by adding dash separators.
QlmFormatLicenseKeyGroupSize	Size of the group of characters in a formatted license. The QlmFormatLicenseKey property must be true for this to take effect.
QlmGUID	Set the GUID associated to your product. The GUID can be found on the Define Product page in the QLM Console.
QlmLicenseType	Set the license type. The license type can be: ComputerName, UserDefined or Generic.
QlmMajorVersion	Set the Major Version associated to your product. The Major Version can be found on the Define Products page in the QLM Console.
QlmMinorVersion	Set the Minor Version associated to your product. The Minor Version can be found on the Define Products page in the QLM Console.
QlmProductID	Set the Product ID Version associated to your product. The Product ID can be found on the Define Products page in the QLM Console.
QlmProductName	Set the Product Name associated to your product.
QlmPublicKey	Set the Public Key associated to your product. The Public Key Version can be found on the Define Products page (Keys tab) in the QLM Console.
QlmStoreKeysOptions	Set whether to store license keys per user, per machine or both.
QlmValidateCertificate	The QLM DLLs are digitally signed by a trusted certificate authority. In order to ensure that hackers do not replace the QLM DLLs by dummy ones, QLM can validate that the DLLs are properly signed.
QlmCommunicationEncryptionKey	Set the communicationEncryptionKey to use when connecting to the The QLM License Server. The communicationEncryptionKey must match the one defined in the web.config file on the web server.
QlmLicenseServerUrl	Set the URL to the QLM License Server. The value of this url is typically: http://yourdomain.com/qlm/qlmservice.asmx

Floating Licenses Share Model

Quick License Manager Enterprise offers all the features of QLM Professional and in addition enables you to implement floating licenses.

QLM Enterprise requires a database at the customer site to manage floating licenses. QLM supports 3 different file formats for the floating license database in varying levels of complexity: XML, MS-Access and SQL Server.

For XML and MS-Access, all that is required on the end user site is a Windows Share that is accessible to all computers on the network. The share must have read/write privileges. On the share, a specific QLM database or XML file must be copied and activated. Below are the steps required to add floating license support to your application:

- Upon purchase, issue an Activation Key to your customer. When creating the activation key, set the FloatingSeats property to the number of seats purchased. If you are issuing the activation key from your eCommerce provider via a URL, set the is_floating argument to true. If you are creating the activation key via the QLM Management Console, set the Floating Seats field to the number of purchased seats.
- Your customer installs your product.
- As part of your product's installation, you should install the QLM Floating License Database on a network share accessible to all users, or on a SQL Server.
- Your customer enters the Activation Key in your product and clicks on Activate.
- Your application connects to the QLM License Server that you are hosting (over the internet) and issues a computer bound key to the customer. The procedure so far is the same as for non floating licenses.
- To initialize floating licenses support, you create an instance of the QlmFloatingLicenseMgr object (for XML) or the QlmFloatingLicenseMgrDb (for MS-Access or SQL Server) object (located in QlmLicense.dll).
- If you are using .NET, call one of the available constructors (except the default constructor).
- If you are using any other language, you need to create an instance of the QlmFloatingLicenseMgr or QlmFloatingLicenseMgrDb class, then set either the QlmLicenseObject or the SettingsFile properties. Check the API reference for more details.
- To initialize the location and password of the QLM Floating xml file or database, call QlmFloatingLicenseMgr.InitializeDb
- To activate the QLM Floating database/xml, you then call QlmFloatingLicenseMgr.RegisterLicense. This registers the license in the floating database with information about the number of available seats. You can use the QLMLicenseWizad.exe to register the license and database with the /floating_master and /floating_node command line arguments.
- When your application starts up, you call the ActivateFloatingLicense method to register the node.
- When your application exits, you call the ReleaseFloatingLicense method to release the node.

A License Viewer application is available and can be distributed with your application (QlmLicenseViewer.exe). The License Viewer enables your customer to view allocated licenses and release them if needed.

A sample showing how to implement floating licenses is provided in the following folder:

%Public%\Documents\Quick License
Manager\Samples\QLMEnterprise\DotNet\C#\QlmEnterpriseSample

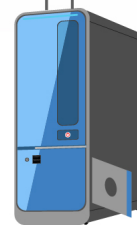
Network share at the customer's site
where the floating license DB is stored



Floating
License DB

3. Initialize the floating license
(InitializeDB/Register License)

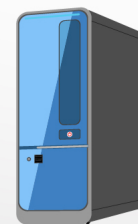
1. Activate purch
license key /



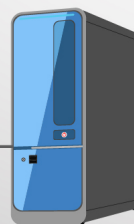
2. Rec

Customer's Server
where your application is installed

4. Activate Floating License when the application satatus
Release Floating License when the appication exist



Customer's Workstation 2



Customer's Workstation 1

Floating Licenses

The QLM Enterprise API includes all QLM Pro methods in addition to methods specific to QLM Enterprise only features. The API Reference section covered here includes the methods that are specific to QLM Enterprise. You can also use any method listed in the API Reference of QLM Pro.

QlmFloatingLicenseMgr and QlmFloatingLicenseMgrDb

This class manages floating licenses on the client's network. It interfaces with an XML file (QlmFloatingLicenseMgr) or an MS-Access/SQL Server database (QlmFloatingLicenseMgrDb).

When the protected application is launched, **ActivateFloatingLicense** should be called to consume a license. When the protected application exits, **ReleaseFloatingLicense** should be called to release the consumed license.

To initialize the QlmFloatingLicenseMgr object, you need to: Initialize the license object Initialize the database location Register the license key

Initialization of the license object can be done in one of 2 ways: (a) by calling one of the constructors that takes the **settingsFile** argument or (b) by calling the constructor that takes a fully initialized QlmLicense object. The xml **settingsFile** must be generated by the **Protect your application** wizard.

Initialization of the database location can be done in one of 2 ways: (a) by calling one of the constructors that takes the dbPath and dbPassword as arguments or (b) by calling the InitializeDb method. Note that the default password that protects the qlmFloating.mdb database is **Coraso23!**

. It is highly recommended that you change this password.

Finally, to register the license key, you must call the **RegisterLicense** method.

Once these 3 steps completed, you can start calling **ActivateFloatingLicense** and **ReleaseFloatingLicense**

.

ActivateFloatingLicense

Activates a floating license. This will consume one license if available. You should call this function when your application is launched. If the function returns `QlmActivationStatus.Activated` or `QlmActivationStatus.AlreadyActivated`, you can proceed with the launch of your application. Any other value would indicate an issue and you should abort the launch of your application.

C#: `QlmActivationStatus ActivateFloatingLicense(string computerID, string computerName, out string message)`

`computerID` - The ID of the computer on which the license should be activated.

`computerName` - The name of the computer on which the license should be activated.

`message` - A returned message with details about the operation.

QlmActivationStatus can have one of the following values:

`Activated` - The license was activated successfully.

`AlreadyActivated` - The license was already activated for this computer.

`FailedToFindDb` - The QLM Floating database was not found.

`FailedToActivate` - The activation failed.

`InvalidLicense` - The license key registered in the database is not valid.

`NoMoreLicenses` - There are no more licenses available.

`NotActivated` - The license was not activated.

`Undefined` - Initial value of this enum.

ConcurrentAccessRetries

This property specifies how many attempts are made to read or write to the floating license xml file before failing. If you expect over 100 concurrent users, you may want to increase this setting. The default is 5.

EnableAutomaticRegistration

When true, the QlmFloatingLicenseMgr class will check on a regular basis if the current node is still registered in the floating license database/xml. If the current running node is no longer registered, it will be automatically re-registered. In the event re-registration is not possible because all the licenses are consumed, a violation event will be triggered.

GetDbData

Gets general settings stored in the floating license xml or database.

C#: bool GetDbData(out string dbPathInDb, out string dbDecryptedPathInDb, out string activationKey, out string computerKey, out string computerID, out string errorMessage)

dbPathInDb - Encrypted value of the full path to the QLM Floating database as registered in the database/xml.

dbDecryptedPathInDb - Non-encrypted full path to the QLM Floating database as registered in the database/xml. If this value does not match the encrypted value, an error will be thrown. The only time this can happen is if the path is manually modified in the xml file.

activationKey - Activation Key.

computerKey- Computer bound license key.

computerID - Identifier of the computer where the license was activated..

errorMessage - returned error message if the function fails.

InitializeDb

Initializes the path and password of the QLM Floating database. This function will throw an exception if the path is not found. The folder where the database is located must be writable by all users.

C#: bool InitializeDb(string dbPath, string dbPassword, bool registerDb, out bool licenseRegistered, out string errorMessage)

dbPath - Full path to the QLM Floating database, including the filename.

dbPassword - Password to open the QLM Floating database.

registerDb - when set to true, the DbPath in the floating license xml will be updated. This should only be set to true once when configuring the xml location.

licenseRegistered - returned flag indicating if the license is already registered in the database.

errorMessage - returned error message if the function fails.

IsFloatingLicenseRegistered

Determines if a floating license is registered. You can call this function randomly in your application to verify if a user did not inadvertently or maliciously release a license that is currently in use. A license can be released by the end user if you give them access to the QLM License Viewer application (QlmLicenseViewer.exe). This function should return QlmActivationStatus.AlreadyActivated. Any other value would indicate an issue.

C#: public QlmActivationStatus IsFloatingLicenseRegistered(string computerID, string computerName, out string message)

computerID - The ID of the computer on which the license should be activated.

computerName - The name of the computer on which the license should be activated.

message - A returned message with details about the operation.

QlmActivationStatus can have one of the following values:

Activated - The license was activated successfully.

AlreadyActivated - The license was already activated for this computer.

FailedToFindDb - The QLM Floating database was not found.

FailedToActivate - The activation failed.

InvalidLicense - The license key registered in the database is not valid.

NoMoreLicenses - There are no more licenses available.

NotActivated - The license was not activated.

Undefined - Initial value of this enum.

QlmFloatingLicenseViolationEvent

At a regular interval, the QlmFloatingLicenseMgr validates that the license of the current node is still registered. This is to address the situation where a node is purposely or inadvertently released by a user via the QLM Floating License Viewer. When a running node determines that it is no longer registered as running, it will automatically attempt to re-register itself. Re-registration will be successful if there is a free license available. If all licenses have been consumed by other nodes, the QlmFloatingLicenseViolationEvent is fired. When this event is fired, you can decide what action to take in your application, for example, quit the application.

C#: event QlmFloatingLicenseEventHandler QlmFloatingLicenseViolationEvent

Example:

```
QlmFloatingLicenseMgr floatingLicense = new QlmFloatingLicenseMgr();
floatingLicense.QlmFloatingLicenseViolationEvent += new
QlmFloatingLicenseEventHandler(OnFloatingLicenseViolationEvent);
public void OnFloatingLicenseViolationEvent(object sender, QlmFloatingLicenseViolationEventArgs e)
{
    MessageBox.Show("License violation detected. Status: " + e.Status);
}
```

QlmLicenseObject

Sets a fully initialize QlmLicense object.

This should typically be set from the constructor, however if the QlmFloatingLicenseMgr class is created via COM instead of C#, you can only invoke the default constructor so you must set this property or the SettingsFile property right after creating the QlmFloatingLicenseMgr object.

Note that you should either set the QlmLicenseObject property or the SettingsFile property but not both.

RegisterLicense

Register the end user license in the qlmFloating DB. This is used to extract the number of floating seats.

C#: `bool RegisterLicense(string activationKey, string computerKey, string computerID)`

Parameters

activationKey - Activation Key

computerKey - The computer key returned as a result of the activation process

computerID - The ID of the computer on which the license was activated. Note that in the context of floating licensing, activation only occurs on one computer. The computerID specified here is the ID of that computer.

ReleaseFloatingLicense

Releases a floating license. You should call this function when you exit your application.

C#: `bool ReleaseFloatingLicense(string computerID, string computerName, out string message)`

`computerID` - The ID of the computer on which the license should be activated.

`computerName` - The name of the computer on which the license should be activated.

`message` - A returned message with details about the operation.

SettingsFile

Set the full path of QLM settings files as generated by the Protect Your Application wizard.

This should typically be set from the constructor, however if the `QlmFloatingLicenseMgr` class is created via COM instead of C#, you can only invoke the default constructor so you must set this property or the `QlmLicenseObject` property right after creating the `QlmFloatingLicenseMgr` object.

Note that you should either set the `QlmLicenseObject` property or the `SettingsFile` property but not both.

ValidateLicenseLocation

Validates that the location of the floating license database matches the value published to the license server.

C#: bool ValidateLicenseLocation(string dbLocation, bool skipValidationIfNoInternet, out QlmActivationStatus activationStatus, out string errorMessage)

Parameters

dbLocation - Path to the floating license database.

skipValidationIfNoInternet - If the system does not have internet access, skip the validation and assume everything is ok.

activationStatus - returns the activation status of the license. Possible values are:

QlmActivationStatus.Undefined, QlmActivationStatus.NotActivated,

QlmActivationStatus.Activated, QlmActivationStatus.InvalidLicense

errorMessage - returned error message in case of a failure

GetFloatingLicenseLocation

Gets the location of the floating license database (or xml file). The location can be set in the QLM database using the SetFloatingLicenseLocation method. In order to ensure that a user has not duplicated the floating license database, you can call SetFloatingLicenseLocation to set the location of the floating license database when it is initially registered. Subsequently, when your application starts up, you can call GetFloatingLicenseLocation and compare the registered location with the real location of the database.

C#: public bool GetFloatingLicenseLocation(string webServiceUrl, string activationKey, out string location, out string message)

Parameters

webServiceUrl - URL to the QLM License Server.

activationKey - activation key associated to this installation.

location - full path of the floating license DB or xml file.

message - return message in case of an error.

SetFloatingLicenseLocation

Sets the location of the floating license database (or xml file). In order to ensure that a user has not duplicated the floating license database, you can call SetFloatingLicenseLocation to set the location of the floating license database when it is initially registered. Subsequently, when your application starts up, you can call GetFloatingLicenseLocation and compare the registered location with the real location of the database.

C#: public bool SetFloatingLicenseLocation(string webServiceUrl, string activationKey, string location, out string message)

Parameters

webServiceUrl - URL to the QLM License Server.

activationKey - activation key associated to this installation.

location - full path of the floating license DB or xml file.

message - return message in case of an error.

Analytics

QLM Enterprise allows you to collect analytics about your application's usage.

In the first release of QLM v7, QLM can collect and reports analytics about installations of your software application.

In subsequent releases, QLM will allow you to collect and report data on feature usage

.

Analytics Installs

To collect data about your application's installs and uninstalls, QLM provides 3 methods: AddInstall, UpdateInstall and RemoveInstall.

AddInstall should be called during the installation of your application or the first time your application runs.

UpdateInstall should be called if any of the data published to the server during installation was modified.

RemoveInstall should be called when your application is uninstalled.

To view reports about your applications installations, start the QLM Application and click on the Analytics tab.

The Analytics tab displays two graphs and a table.

The Trial Installs

graph displays statistics about all the trial installs and uninstalls of your application.

The Permanent Installs

graph displays statistics about all the permanent installs and uninstalls of your application.

The All Installs

grid displays data about all installs and uninstalls.

.

AddInstall

public bool AddInstall(string softwareVersion, string osVersion, string computerName, string computerID, string activationKey, string computerKey, bool trial, string productName, int majorVersion, int minorVersion, ref string installID).

Description

Registers an installation with the server. You should call this function once when your application is installed. You should store the returned installID in your application's settings and reuse it on subsequent calls to the QlmAnalytics API.

Parameters

softwareVersion: Version of your software
osVersion: Version of the operating system
computerName: Name of the computer
computerID: Unique identifier of the computer
activationKey: activation key on the system
computerKey: computer key associated to the system
productName: name of your product
majorVersion: major version of your product
minorVersion: minor version of your product
installID: unique identifier of this installation, returned from the server.

Return

Returns true if the data was successfully published to the server.

AddInstall

```
public bool AddInstall(string softwareVersion, string osVersion, string computerName, string  
computerID, string activationKey, string computerKey, bool trial, string productName, int majorVersion,  
int minorVersion, string customData1, string customData2, string customData3, ref string installID).
```

Description

Registers an installation with the server. You should call this function once when your application is installed. You should store the returned installID in your application's settings and reuse it on subsequent calls to the QlmAnalytics API.

Parameters

softwareVersion: Version of your software
osVersion: Version of the operating system
computerName: Name of the computer
computerID: Unique identifier of the computer
activationKey: activation key on the system
computerKey: computer key associated to the system
productName: name of your product
majorVersion: major version of your product
minorVersion: minor version of your product
customData1: your own custom data
customData2: your own custom data
customData3: your own custom data
installID: unique identifier of this installation, returned from the server.

Return

Returns true if the data was successfully published to the server.

UpdateInstall

public bool UpdateInstall(string installID, string softwareVersion, string osVersion, string computerName, string computerID, string activationKey, string computerKey, bool trial, string productName, int majorVersion, int minorVersion).

Description

Updates information of a registered installation on the server.

Parameters

installID: unique identifier of this installation, returned by a call to AddInstall
softwareVersion: Version of your software
osVersion: Version of the operating system
computerName: Name of the computer
computerID: Unique identifier of the computer
activationKey: activation key on the system
computerKey: computer key associated to the system
productName: name of your product
majorVersion: major version of your product
minorVersion: minor version of your product

Return

Returns true if the data was successfully published to the server.

UpdateInstall

public bool UpdateInstall(string installID, string softwareVersion, string osVersion, string computerName, string computerID, string activationKey, string computerKey, bool trial, string productName, int majorVersion, int minorVersion, string customData1, string customData2, string customData3).

Description

Updates information of a registered installation on the server.

Parameters

installID: unique identifier of this installation, returned by a call to AddInstall
softwareVersion: Version of your software
osVersion: Version of the operating system
computerName: Name of the computer
computerID: Unique identifier of the computer
activationKey: activation key on the system
computerKey: computer key associated to the system
productName: name of your product
majorVersion: major version of your product
minorVersion: minor version of your product
customData1: your own custom data
customData2: your own custom data
customData3: your own custom data

Return

Returns true if the data was successfully published to the server.

UpdateLastAccessedDate

public bool UpdateLastAccessedDate(string installID).

Description

Updates the Last Accessed Date associated to a registered installation on the server.

Parameters

installID: unique identifier of this installation, returned from the server.

Return

Returns true if the data was successfully published to the server.

RemoveInstall

```
public bool RemoveInstall(string installID, out string errorMessage)
```

Description

Unregisters an application with the server. You should call this function when the user uninstalls your application.

Parameters

installID: unique identifier of this installation, returned from the server.

errorMessage: returned error message if the call fails.

Return

Returns true if the data was successfully published to the server.

Protect Android Apps

QLM Pro can protect Android applications with permanent, trial and device bound keys. A java package (source code included) exposes an API that enables you to provide your customers with an evaluation of your software and then turn it at anytime into a permanent license, or simply extend it for subscription based applications. The java package along with a sample applications are provided in the following QLM Pro samples folder:

`%Public%\documents\quick license manager\samples\qlmenterprise\Android\Qlm.Vendor.App`

The sample contains 2 packages: `com.soraco.qlm` and `com.vendor.app`

com.soraco.qlm

is the package that performs the license validation, activation, encryption, etc. You typically do not need to change any code in this package. The `QlmLicense` class in this package is the main class you need to interact with. The most common methods of this class are document in the help under "Quick License Manager Professional / API reference / Mobile Devices API".

com.vendor.app

simulates your application. When the application is launched, the `ValidateActivity` class checks if a license has ever been activated on the device. If no key was ever activated, the `PrefsActivity` is started to allow the user to enter an Activation Key and activate it.

Use the QLM Pro Application to create an Activation Key from the Manage Keys tab. Note that activation keys can be created from your server using our API or directly from one of the ecommerce provider integrated with QLM. For a full list of ecommerce providers integrated with QLM, check the help or our web site.

Once the user enters the Activation Key, the `PrefsActivity` class calls the `QlmLicense.ActivateLicense` method. If activation is successful, encrypted license information is stored on the device. QLM uses RSA asymmetric encryption to store license information on the device. The RSA public/private key pair is automatically generated by QLM when you define a product in the QLM Application Define Products page. The keys are displayed on the Encryption Keys tab / Mobile Devices Encryption.

Note that the encrypted data stored on the device is encrypted on the QLM server using the RSA private key and decrypted on the device using the RSA public key.

On the Android application, the RSA public key must be stored in a file called `QlmPublicKey.xml` in the assets folder.

In the event a client does not have an internet connection to activate a license online, you can perform an **offline activation**

as described below:

- In the QLM application, under the Manage Keys tab, locate and select the license to activate.
- Click on the Activate button.
- Fill in the Computer ID field on the Activation tab along with other fields as required.
- Click on the **Mobile Device Activation** tab.
- Select a location where you would like to store a license file then click Ok.
- Send the generated license file to your customer and ask them to copy it to the folder where your application expects the license file to be located.

Protect .NET Android Apps

QLM Pro can protect .NET / Xamarin applications running on Android devices with permanent, trial and device bound keys.

A .NET library (source code included) exposes an API that enables you to provide your customers with an evaluation of your software and then turn it at anytime into a permanent license, or simply extend it for subscription based applications. The .NET library along with a sample applications are provided in the following QLM Pro samples folder:

```
%Public%\documents\quick license  
manager\samples\qlmenterprise\Android\DotNet\QlmDotNetAndroidSample  
The sample contains 3 project: QlmMonoAndroidLib, QlmXamLicenseLib and  
QlmDotNetAndroidSample
```

QlmMonoAndroidLib

is the library that contains .NET Mono classes required for decrypting license information sent by the QLM License Server.

QlmXamLicenseLib

is the library that performs the license validation, activation, decryption, etc. You typically do not need to change any code in this package. The QlmLicense class in this package is the main class you need to interact with. The most common methods of this class are document in the help under "Quick License Manager Professional / API reference / Mobile Devices API".

QlmDotNetAndroidSample

simulates your application. When the application is launched, the application attempts to retrieve a stored license on the device to validate it. If no key was ever activated, the user is prompted to enter an Activation Key and activate it.

Use the QLM Pro Application to create an Activation Key from the Manage Keys tab. Note that activation keys can be created from your server using our API or directly from one of the ecommerce provider integrated with QLM. For a full list of ecommerce providers integrated with QLM, check the help or our web site.

Once the user enters the Activation Key, you call the QlmLicense.ActivateLicense method. If activation is successful, digitally signed license information is stored on the device. QLM uses RSA asymmetric encryption to store license information on the device. The RSA public/private key pair is automatically generated by QLM when you define a product in the QLM Application Define Products page. The keys are displayed on the Encryption Keys tab / Mobile Devices Encryption.

Note that the encrypted data stored on the device is signed on the QLM server using the RSA private key and verified on the device using the RSA public key.

In the QlmDotNetAndroidSample application, the RSA public key is stored in a file called QlmPublicKey.xml. In your own application, it is recommended that you hard code the public key in your code rather than store it in an external file.

In the event a client does not have an internet connection to activate a license online, you can perform an **offline activation**

as described below:

- In the QLM application, under the Manage Keys tab, locate and select the license to activate.
- Click on the Activate button.
- Fill in the Computer ID field on the Activation tab along with other fields as required.
- Click on the **Mobile Device Activation** tab.
- Select a location where you would like to store a license file then click Ok.
- Send the generated license file to your customer and ask them to copy it to the folder where your application expects the license file to be located.

Protect iOS Apps

QLM Pro can protect iOS applications with permanent, trial and device bound keys. An Objective-C library (source code included) exposes an API that enables you to provide your customers with an evaluation of your software and then turn it at anytime into a permanent license, or simply extend it for subscription based applications. The Objective-C library along with a sample applications are provided in the following QLM Pro samples folder:

%Public%\documents\quick license manager\samples\qlmenterprise\iOS\QlmiOSSample

The sample contains 2 projects: QlmLicenseMobile and QlmMobileDeviceSample.

QlmLicenseMobile

is the project that performs the license validation, activation, encryption, etc. You typically do not need to change any code in this package. The QlmLicense class in this package is the main class you need to interact with. The most common methods of this class are document in the help under "Quick License Manager Professional / API reference / Mobile Devices API".

QlmMobileDeviceSample

simulates your application. When the application is launched, the application startup code checks if a license has ever been activated on the device. If no key was ever activated, the user is required to enter an Activation Key and activate it. If a key was previously activated, the license is validated to determine if the license is still valid and has not expired.

Use the QLM Pro Application to create an Activation Key from the Manage Keys tab. Note that activation keys can be created from your server using our API or directly from one of the ecommerce provider integrated with QLM. For a full list of ecommerce providers integrated with QLM, check the help or our web site.

Once the user enters the Activation Key, you call the QlmLicense.ActivateLicense method with the required arguments. If activation is successful, encrypted license information is stored on the device. QLM uses RSA asymmetric encryption to store license information on the device. The RSA public/private key pair is automatically generated by QLM when you define a product in the QLM Application Define Products page. The keys are displayed on the Encryption Keys tab / Mobile Devices Encryption.

Note that the encrypted data stored on the device is encrypted on the QLM server using the RSA private key and decrypted on the device using the RSA public key.

On the iOS application, the RSA public key should ideally be embedded in your application. The sample code loads the public key from an external file called QlmPublicKey.xml but it is recommended that the public key be hard coded in your applicaiton.

In the event a client does not have an internet connection to activate a license online, you can perform an **offline activation**

as described below:

- In the QLM applicatinon, under the Manage Keys tab, locate and select the license to activate.
- Click on the Activate button.
- Fill in the Computer ID field on the Activation tab along with other fields as required.
- Click on the **Mobile Device Activation** tab.
- Select a location where you would like to store a license file then click Ok.
- Send the generated license file to your customer and ask them to copy it to the folder where your application expects the license file to be located.

Protect Mac OS X Apps

QLM Pro can protect Mac OS X applications with permanent, trial and device bound keys. An Objective-C library (source code included) exposes an API that enables you to provide your customers with an evaluation of your software and then turn it at anytime into a permanent license, or simply extend it for subscription based applications. The Objective-C library along with a sample applications are provided in the following QLM Pro samples folder:

%Public%\documents\quick license manager\samples\qlmenterprise\Mac\QlmMacSample

The sample contains 2 projects: QlmLicenseMobile and QlmMobileDeviceSample.

QlmLicenseMobile

is the project that performs the license validation, activation, encryption, etc. You typically do not need to change any code in this package. The QlmLicense class in this package is the main class you need to interact with. The most common methods of this class are document in the help under "Quick License Manager Professional / API reference / Mobile Devices API".

QlmMobileDeviceSample

simulates your application. When the application is launched, the application startup code checks if a license has ever been activated on the device. If no key was ever activated, the user is required to enter an Activation Key and activate it. If a key was previously activated, the license is validated to determine if the license is still valid and has not expired.

Use the QLM Pro Application to create an Activation Key from the Manage Keys tab. Note that activation keys can be created from your server using our API or directly from one of the ecommerce provider integrated with QLM. For a full list of ecommerce providers integrated with QLM, check the help or our web site.

Once the user enters the Activation Key, you call the QlmLicense.ActivateLicense method with the required arguments. If activation is successful, encrypted license information is stored on the device. QLM uses RSA asymmetric encryption to store license information on the device. The RSA public/private key pair is automatically generated by QLM when you define a product in the QLM Application Define Products page. The keys are displayed on the Encryption Keys tab / Mobile Devices Encryption.

Note that the encrypted data stored on the device is encrypted on the QLM server using the RSA private key and decrypted on the device using the RSA public key.

On the OS X application, the RSA public key should ideally be embedded in your application. The sample code loads the public key from an external file called QlmPublicKey.xml but it is recommended that the public key be hard coded in your applicaiton.

In the event a client does not have an internet connection to activate a license online, you can perform an **offline activation**

as described below:

- In the QLM applicatinon, under the Manage Keys tab, locate and select the license to activate.
- Click on the Activate button.
- Fill in the Computer ID field on the Activation tab along with other fields as required.
- Click on the **Mobile Device Activation** tab.
- Select a location where you would like to store a license file then click Ok.
- Send the generated license file to your customer and ask them to copy it to the folder where your application expects the license file to be located.

Protect Java Desktop Apps

QLM Pro can protect Java desktop applications with permanent, trial and device bound keys. A java package (source code included) exposes an API that enables you to provide your customers with an evaluation of your software and then turn it at anytime into a permanent license, or simply extend it for subscription based applications. The java package along with a sample applications are provided in the following QLM Pro samples folder:

```
%Public%\documents\quick license  
manager\samples\qlmenterprise\JavaDesktop\QlmJavaDesktopSample
```

The sample contains 2 packages: qlmLicenseMobile and qlmMobileDeviceSample.

qlmLicenseMobile

is the package that performs the license validation, activation, decryption, etc. You typically do not need to change any code in this package. The QlmLicense class in this package is the main class you need to interact with. The most common methods of this class are document in the help under "Quick License Manager Professional / API reference / Mobile Devices API".

qlmMobileDeviceSample

simulates your application. When the application is launched, the code checks if a license has ever been activated on the device. If no key was ever activated, a dialog is launched to allow the user to enter an Activation Key and activate it.

Use the QLM Pro Application to create an Activation Key from the Manage Keys tab. Note that activation keys can be created from your server using our API or directly from one of the ecommerce provider integrated with QLM. For a full list of ecommerce providers integrated with QLM, check the help or our web site.

Once the user enters the Activation Key, the code calls the QlmLicense.ActivateLicense method. If activation is successful, encrypted license information is stored on the system. QLM uses RSA asymmetric encryption to store license information on the system. The RSA public/private key pair is automatically generated by QLM when you define a product in the QLM Application Define Products page. The keys are displayed on the Encryption Keys tab / Mobile Devices Encryption.

Note that the encrypted data stored on the device is encrypted on the QLM server using the RSA private key and decrypted on the user system using the RSA public key.

In the event a client does not have an internet connection to activate a license online, you can perform an **offline activation**

as described below:

- In the QLM applicatinon, under the Manage Keys tab, locate and select the license to activate.
- Click on the Activate button.
- Fill in the Computer ID field on the Activation tab along with other fields as required.
- Click on the **Mobile Device Activation** tab.
- Select a location where you would like to store a license file then click Ok.
- Send the generated license file to your customer and ask them to copy it to the folder where your application expects the license file to be located.

Android API

The Android API provides classes that you can use from your Android application to validate or activate license keys.

Note that all methods exposed by the QLM License Server cannot be called directly via SOAP. In order to communicate with the QLM License Server, you need to use the QLM Android API methods that are exposed via the `com.soraco.qlm` package.

ActivateLicense

Activates a license by connecting to a QLM License Server

boolean ActivateLicense(String webServiceUrl, String activationKey, String deviceID, String computerKey, QlmResult qlmResult)

webServiceUrl: URL to the QLM License Server. Example:

<https://qlm3.net/qlmdemov16/QlmLicenseServer/qlmservice.aspx>

activationKey: license key to activate, typically entered by the user in a license registration form.

deviceID: Unique identifier of the device.

computerKey: computer bound license key. This key is typically empty for the first activation. Upon successful activation, a computer key is returned by the QLM License Server and stored on the device.

qlmResult: class that returns the result of the activation

Return

True if the activation succeeded.

False if the activation failed.

getStatus

Returns the last status. See ELicenseStatus for possible values.

int getStatus ()

Return

Validation status of the license key.

getExpiryDate

Returns the expiry date of the evaluation key.

Date getExpiryDate()

Return

Expiry date of the evaluation key.

getRemainingDays

Returns the number of days remaining in the trial.

int getRemainingDays ()

Return

Days remaining in the trial.

IsLicenseValid

Checks if a license is valid

boolean IsLicenseValid()

Return

True if the license is valid.

False if the license is not valid.

QLM Portal

The QLM Portal provides a web interface for managing license keys.

To access the portal, you need to create a user account. User Accounts are created from the QLM Management Console under Manage Keys / User Accounts. Your QLM Portal license entitles you to create a single user account. If you would like to create additional accounts for internal use or for your affiliates, you need to purchase QLM Portal User Licenses from our web site. Each User Account is associated to a User Profile. User Profiles determine what users can view, create or modify via the QLM Portal. A User Accounts associated with the built-in User Profile "None" can perform any action and view all data. This is equivalent to an Administrator account You can configure the following restrictions for User Profiles:

- Maximum number of trial keys per system
- Maximum number of permanent keys per system
- Maximum total keys
- Maximum activations per key

In addition, you can control what operations a user can perform from the QLM Portal:

- Creating new keys
- Activating keys
- Releasing keys
- Deleting keys
- Creating customers
- Deleting customers
- Exporting keys
- Setting the Expiry Duration of a new key
- Settings the Expiry Date of a new key
- Settings the Maintenance Plan option
- Setting the Generic license option

QLM Portal

The QLM Portal provides a web interface for managing license keys.

To access the portal, you need to create a user account. User Accounts are created from the QLM Management Console under Manage Keys / User Accounts. Your QLM Portal license entitles you to create a single user account. If you would like to create additional accounts for internal use or for your affiliates, you need to purchase QLM Portal User Licenses from our web site. Each User Account is associated to a User Profile. User Profiles determine what users can view, create or modify via the QLM Portal. A User Accounts associated with the built-in User Profile "None" can perform any action and view all data. This is equivalent to an Administrator account You can configure the following restrictions for User Profiles:

- Maximum number of trial keys per system
- Maximum number of permanent keys per system
- Maximum total keys
- Maximum activations per key

In addition, you can control what operations a user can perform from the QLM Portal:

- Creating new keys
- Activating keys
- Releasing keys
- Deleting keys
- Creating customers
- Deleting customers
- Exporting keys
- Setting the Expiry Duration of a new key
- Settings the Expiry Date of a new key
- Settings the Maintenance Plan option
- Setting the Generic license option

Installing the QLM Portal

The QLM portal can be installed via the setup (QlmLicenseServerSetup.exe) or manually.

If you can run a setup program on your web server, run the QlmLicenseServerSetup.exe on your server and make sure that the QLM Portal feature is selected.

If you cannot run a setup program on your web server, following are the manual steps to install the QLM Portal:

- Create a virtual directory on your web server called: QlmPortal
- Upload all the files in the %Public%\Public Documents\Quick License Manager\DeployToServer\QlmPortal folder to the QlmPortal virtual directory
- Enable ASP.NET 4.0 for the virtual directory.
- Customize the following appSettings in the web.config file as follows:
 - communicationEncryptionKey
 - adminEncryptionKey
 - sqlSyntax
 - webServiceUrl
- Additionally, you should customize the connectionStrings settings in the web.config file to the same values as the web.config of the QLM License Server.

The QLM portal uses ASP.NET forms authentication to validate users. To enable ASP.NET forms authentication, the QLM database needs to be updated to maintain authentication information. If you have recently purchased or are evaluating QLM, this step is already performed during the normal setup. If you are upgrading from QLM v6 or earlier, you must run the sql2005.aspnet.sql script located in the following folder:

%Public%\Public Documents\Quick License Manager\DeployToServer\QlmLicenseServer\Db

Registering your license

To register your license and enable the QLM Portal add-on:

- Start the QLM Application
- Go to the Manage Keys tab
- Click on Sites
- Select your License Server profile
- In the Portal License Key field, enter your QLM Portal license key and click on Register. Note that the QLM Portal License Key is not the same as your QLM Professional or Enterprise license key. If you have not purchased the QLM Portal add-on, you can do so from our web site.

Accessing the QLM Portal

The first step in setting up a portal is to create the Administrator's user account. To do so, start the QLM Application and:

- Go to the Manage Keys tab.
- Click on User Accounts in the Portal group.
- Click on the Add button to create a new user account.
- Fill in all the fields as required. When asked to associate a User Profile with the user account, if you select None, the user will be able to create an unlimited amount of keys. If you select a User Profile, the user will be an Administrator.

Now that you have an account, you may login to the portal at the following URL:

<http://yourserver/qlmportal/qlmportal.aspx>

To create a user account for your end-users or affiliates:

- Go to the Manage Keys tab.
- Click on User Profiles in the Portal group.
- Click on the Add button to create a new User Profile. You can control how many license keys the user can create, for which products they can create license keys and what operations they are allowed to perform.
- Once the User Profile is created, click on User Accounts in the Portal group.
- Click on the Add button to create a new user account and select the corresponding User Profile from the User Profile dropdown.

On the QLM Portal, users can only see customers that they are associated to. If a customer is created from the QLM Portal, the customer is automatically associated to the logged in user profile. If a customer is created from the QLM Management Console / Manage Customers tab, you can associate the customer to a User Profile when the customer is created or at any other time by clicking on the Manage Customers / Edit option.