Software License Models

A white paper

Reprise Software, Inc.

What exactly is a *License Model*?

Put most simply, a *License Model* is a set of terms and conditions which your license manager enforces. For any given set of terms and conditions (i.e., the particular *License Model*), your company has a set of pricing guidelines for the sale of the product.

Let's take a simple example. Your company may sell your software in two different ways – a floating (concurrent use) license for $3000, and a node-locked license for $1200. In this case, the *License Model* could be called either floating or node-locked.

But your *License Model* can (and often is) much more complicated. For example, you might sell a floating license which works only in certain timezones. You might also sell a more unrestricted floating license, which operates in any timezone. In this case, both *License Models* are of the basic floating type, but the other restrictions define the differences in the *License Models*. Your company might now call these "Floating-timezone-restricted" and "Floating-unrestricted".

> It is important to your development organization that changes to the license model do not result in code changes. This is also important to sales and marketing, who will want to try different License Model offers without having to wait for a new software release.

This white paper discusses a number of common License Models, and with each one, we will list the Reprise License Manager (RLM) license attributes that are used to implement it. Don't worry if you do not understand the RLM nomenclature at this point, the important thing is to understand the varieties of licenses you can deliver with a license manager and the choices you would make in order to implement that model.

Floating License

The floating license is what made license managers famous. All license managers support this. The idea is that some specified number of independent instances of the application can be run anywhere on your customer's network so long as that number does not exceed a predefined limit (the limit in the license file).

In RLM ⇨ every license has a *count*, and if the count is non-zero, this is a floating license for that many instances of the application. The license itself has no associated hostid, meaning that it will run anywhere. The license server (specified by the HOST and ISV lines) keeps track of the # of instances in use.

Node-locked License

A node-locked license is a license grant which allows the software to be used on a particular computer, and on that computer only. Most typically, this license is uncounted, meaning that if the software is running on the specified computer, any number of instances are allowed to execute.

In RLM ⇨ set the *count* field of the license to "uncounted" or "0", and specify the *hostid* of the computer in the actual license. Typically, node-locked uncounted licenses do not require a license server, so they are very simple to deploy.

Node-locked, Limited License

A variant of the node-locked uncounted license, it is sometimes desirable to allow only a limited number of instances of the software to run on a particular computer. This is a counted license which is also node-locked.

In RLM ⇨ set the *count* field of the license to a non-zero value (or ***single***), and specify the *hostid* of the computer in the actual license. This license requires a license server (HOST and ISV lines), unless you use a ***single*** license.

Shared Floating License

Sometimes it is desirable to modify the behavior of a floating license so that all invocations of your product on a single computer use only one license. Or all invocations by the same user. Or all invocations from a particular process tree. In this case, license managers provide a method of sharing a particular license among multiple instances of the product.

In RLM ⇨ Use a floating license and specify the *share=* attribute in the actual license. *share=u* will cause all invocations from a particular user to share the license. *share=uh* will cause all invocations from a particular user on a particular machine to share the license. The optional (:count) at the end of the share specification will allow you to share a license only among a certain number of instances; the next instance will require an additional license. (e.g, *share=u:4* will allow a particular user to run 4 copies using a single license, but the 5th copy will use an additional license.) This license requires a license server (HOST and ISV lines).

Named-User License

A named-user license allows a limited number of users access to a floating license. This allows you to sell a number of instances of your software to a subset of users at the customer site, without having to identify the users at the time you create the license.

In RLM ⇨ Use a floating license and specify the *named_user* or *named_user=n* attribute in the actual license. If you specify *named_user*, then your customer can assign as many users as there are licenses available. Specifying *named_user=n* will allow you to set a lower (or higher) limit for the number of users. This license requires a license server (HOST and ISV lines), unless you use a ***single*** license.

Metered License

A metered license allows you to allocate usage and consume the allocation as something happens in your software. For example, you could meter the number of times your program is run (as opposed to the *concurrent limit of execution* in a floating license). Or you could count the number of pages printed in a word-processing application. Metering also allows you to create a license which will run for a predetermined amount of running time (program running time, as opposed to an expiration date – expiration dates are usually available for all license types).

In RLM ⇨ set the *count* field of the license to ***meter***, and specify the metering parameters in the actual license. The metering parameters control how usage is consumed when you call *rlm_checkout()* and subsequently as the application continues to run. This license requires a license server (HOST and ISV lines), unless you use a ***single*** license.

Token-Based License

A token-based license defines the license you request in terms of other licenses (called the *primary license* or *licenses*) which were issued to your customer. This can be used for several different purposes:

- Allows you to define several licenses in terms of a single, common primary license. Your customer can purchase many copies of the primary license, then they are allowed to run whatever products are defined to use that license. A big advantage is as you release new products which use the same primary license, your customer can use these products immediately, creating more contention for the licenses, and additional sales for you.
- Allows you to create product *bundles* or *packages*.
- Allows you to provide a mapping from a particular license request to one or several equivalent licenses. Typically, ISVs are happy to allow more expensive licenses to be used to satisfy the request of a lower-cost product. This allows your customers to keep working while increasing contention for the higher-priced licenses.

In RLM ⇨ set the *count* field of the license to ***token***, and specify the primary license(s) which are used to satisfy the request for the product. The token license is generally the same for all your customers, and you issue licenses of the *primary* license when they purchase your software. This license requires a license server (HOST and ISV lines), unless you use a ***single*** license.

"Maintenance-Thru-Date" License

Many ISVs wish to issue a license to their customer which allows the customer to run (forever) any version of the software which is released through a particular date, e.g. one year into the future. If the ISV releases a new version in 11 months, the customer can use this version as well, but no version which is released more than 12 months later. This is accomplished by what we call a "date-based" version.

In RLM ⇨ set the *version* field of the license to ***a date, in the format yyyy.mm***, and specify the release date in your call to rlm_checkout() in the same format. When you issue licenses, issue them with a version number corresponding to the expiration of their support. So, for example, if you want to issue one-year supported licenses, in May of 2013, you would issue licenses of version 2014.05. When you release your software in December of 2013, you would request version 2013.12 Note that while it is possible to use other date formats, the format above is used by RLM Activation Pro

About Reprise Software

Reprise Software was founded in 2006 by the creators of FLEXlm. We develop software licensing products, and software licensing products only. Benefit from our 28+ years of experience in the field of license management to have an 'it just works' experience. At Reprise we constantly focus on making our products easier to install and use, so you can have licensing as a benefit - not as an obstacle – to your daily work. Reprise Software is driven to make RLM the best software licensing system available while helping lower the total cost of ownership of licensed applications for both software vendors and their customers.

Rev 1.1 August 2016

