

Improving Customer Satisfaction by Improving Licensing Implementation Quality

Dan Griffith - Manager, Comprehensive Software Asset Management, Motorola
Rich Kline - Strategy Consultant, Flexera Software



Improving Customer Satisfaction by Improving Licensing Implementation Quality

Executive Summary

Thousands of software and embedded software vendors have successfully implemented Flexera Software's FlexNet® licensing solution to reduce unlicensed software usage, enable new customer centric licensing models and improve their customers' ability to manage their license assets. To enable successful licensing implementations in thousands of diverse applications, Flexera Software has provided a very robust and flexible licensing capability. To date, however, Flexera Software has not provided any best-practice guidelines for implementing licensing. As a result, the quality of licensing implementations across different vendors varies widely from the end user's perspective.

This white paper, jointly authored by Flexera Software, named by IDC as the industry leader in software licensing, and Motorola, a prominent member of the licensing user community, seeks to educate vendors on how to improve the quality of licensing implementations by providing a summary of the common issues end users face, and by outlining key implementation considerations vendors should bear in mind. Finally, this white paper introduces the FlexNet Certification Program, which is designed to give vendors a low-cost way to validate and certify the quality of their FlexNet licensing implementations.

Quality of Licensing Implementations

Background

A majority of Flexera Software's large corporate customers have implemented, or are in the process of implementing, global software licensing infrastructures. These infrastructures enable large enterprise customers to share licenses across geographically and organizationally fragmented user communities.

Many companies have implemented centralized procurement and license management on behalf of their end user communities, flexing their corporate muscle to obtain enterprise

license deals and off price-book licensing models, terms and pricing. They are also providing much of the market impetus behind utility based pricing and software as a service.

Because global licensing infrastructures are increasingly being used to deliver FLEXenabled applications, the differences between high and low quality licensing implementations are becoming more visible and frustrating to system administrators running these networks. At the same time, administrators are under increasing pressure to provide greater granularity in reporting back to business and budget owners.

Server-Based License Model

Common issues faced by license administrators within the enterprise environment include:

- License files that are very difficult to trace back to the product that licenses were ordered and purchased for
- Applications whose license usage is incredibly difficult, and in some cases impossible, to report accurately
- Applications with complex license dependencies
- Applications that do not behave in a logical customer friendly manner when a license expires or otherwise becomes unavailable

This white paper provides recommendations on the procurement, reporting and dependency issues commonly encountered today.

Licensing Implementation Recommendations

Treat Licensing as a "Tier One" Product Component

Treat licensing as you would any other product component. Design and incorporate your licensing implementation before product testing begins. This methodology ensures that your licensing scheme is not an afterthought and can be properly tested.

A side benefit of this approach is more effective testing. Reviewing license usage reports from beta program testers can show how the licensed products are actually being used, information that is not otherwise available. Issuing node-locked, expiring licenses during the testing cycle ensures that your newest products do not end up in the hands of competitors.

Licensing application versions so they expire at the end of the beta period also ensures that customers upgrade to your final release, reducing on-going support costs.

Establish a Licensing Architect Function

Flexera Software recommends that vendors designate a licensing architect responsible for driving consistent implementation across multiple products and product lines. License administrators dislike it when products from the same vendor have different failover policies or different ways of defining and enforcing an essentially equivalent unit of usage.

Ensure Licensed Feature Usage Can Be Reported Inside a Product Context

Simply put, provide customers with the ability to report on the total duration a product is in use, as well as the ability to report on licensed feature usage within a product.

This is best accomplished if there is a primary feature name that follows these guidelines.

- It must clearly identify the product it enables.
- It must be checked out every time the product is run, and must remain checked out the entire time the product is in use.
- It must be unique and cannot be shared with or utilized in other products. During the creation of new products or the bundling of products, the primary feature name of a previous product must not be utilized in the licensing of the new product or bundle

To provide additional guidance on how a customer typically would wish to report on license usage, consider the following product family with both core and optional licensed features: *Note that in order to run Graph 10, the Chart 10 license needs to be initially served by a license server. The same is true for Present 10 and Tool Suite 10. Because two licenses are required, customers can report on the overall product usage as well as on the usage of secondary feature components. This is particularly important in the case of the Tool Suite 10 product, where the individual licenses for Write 10, Present 10 and Count 10 may be checked out and returned many times, but there is always at least one component in use and so one Tool Suite 10 license being consumed.*

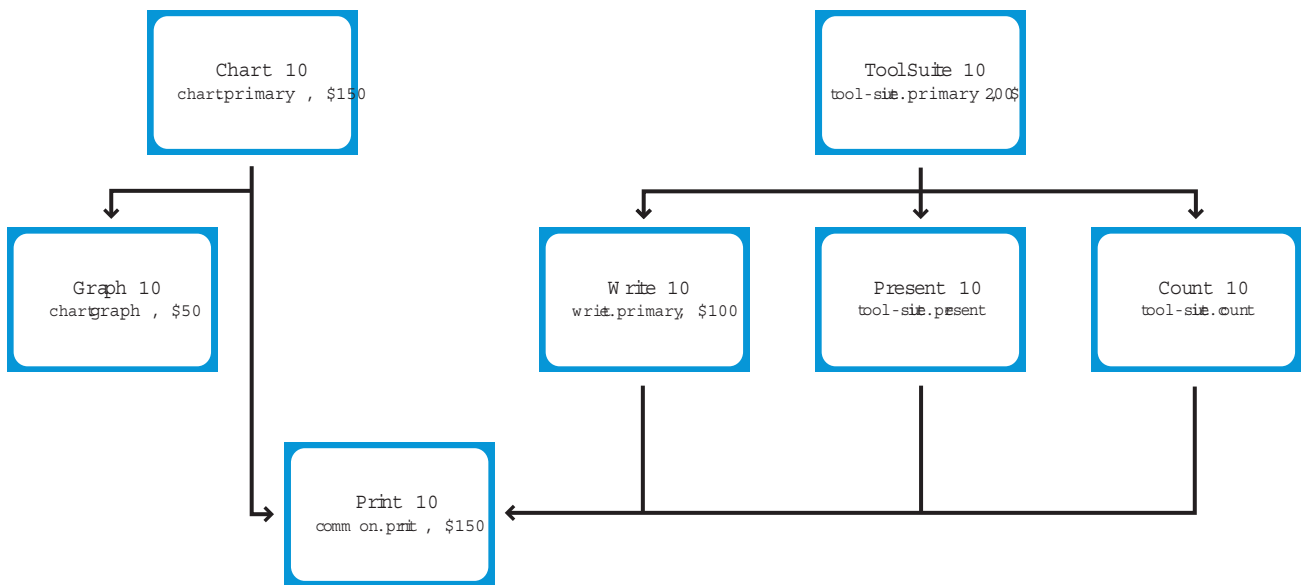


Diagram 1 shows products Chart 10 and Tool Suite 10, where Tool Suite 10 is a shell application that calls the product components Write 10, Present 10 and Count 10.

Secondary features are used to license optional components of your application. An example would be where a customer purchases Chart 10 for \$150 (see Diagram 1) and also purchases the Graph Module for \$50. A sub-feature can be used to provide access to the Graph functionality. This license can be provided to the customer either during the initial purchase or at later date when the customer requires the functionality.

Common features are used to license software components shared between two or more of your primary applications. By properly identifying the components as common and recording which applications are using them (see section entitled "Utilize FlexNet Checkout Data to Provide Product Context"), customers are able to create accurate usage reports.

Using FlexNet's PACKAGE Capabilities

Expanding on our previous example, what if Tool Suite 10 is not a product, but a marketing package of the stand-alone products Write 10, Present 10 and Count 10? This marketing package is not statically determined when releasing the products, but may be dynamically created in the license file, even after products are already installed at the customer location.

In this case, it is possible to report on the usage of the overall Tool Suite 10 package, as well as the individual components of this marketing package, by constructing the Tool Suite 10 package in the license file using the:

```
SUITE_RESERVED option.
PACKAGE Tool_Suite_10 vendorname 10.0 \
COMPONENTS = "write.primary present.primary \
count.primary" OPTIONS=SUITE_RESERVED
```

This enables users to report on both the package usage as well as the individual component usage under the package.

Implement Consistent Feature Naming Conventions

Vendor daemon names

Vendor daemons should be chosen so that a customer is easily able to identify and locate the software publisher.

Feature naming convention

Our recommended feature naming convention is as follows:
application_name.feature_type

Example syntax for the primary feature of the Chart 10 Product in Diagram 1 is as follows:

```
chart.primary
```

Example syntax for Chart 10's sub-features is as follows:

```
chart.graph
chart.pivot (not shown)
```

Example syntax for common feature naming:

```
common.print
common.export
```

Implementing a well thought-out naming convention allows the product and feature name in a single license file line to be understood regardless of the surrounding context. This becomes important when you have lots of products or support multiple licensing models (e.g. node-locked and floating licenses).

Abbreviations should **not** be used for feature names, unless they are easily associated with the feature and clearly spelled out in the product documentation and searchable in the on-line help files.

Meaningless abbreviations contribute to the reporting problems found in many licensing implementations today.

Let us consider a feature called Chip Compiler

Acceptable abbreviation:

ChpCom, ChpComp, ChipComp etc.

Unacceptable abbreviation:

ChpOpt

Finally, vendors should make every effort to keep feature naming standards intact throughout the life of a product. Mid-life product name changes can substantially disrupt license usage reports and processes. See our related comments below under "Feature Naming and Application Versioning."

Implementation Note: Feature Naming and Licensing Security Flexera Software recommends that the feature name passed into the checkout routine be assembled in dynamic memory. Feature names left in static memory can compromise security in two ways. First, they can be easily found in an application binary and changed by anyone with a rudimentary software engineering background. For example, a customer with a valid license for one product could activate another product simply by changing the second product's feature name to match the one on his valid license. In this case the feature name can only be changed to one that has at least the same number of characters.

Second, feature names left in static memory provide a target for software pirates trying to find the embedded FlexNet licensing code in the application. If the checkout call has been found in the code, it is possible to disable licensing. Admittedly, this does take more than just a rudimentary understanding of software and a certain amount of determination.

Consequently, vendors should dynamically build all the components of feature names to ensure that feature naming does not become the weakest link in the chain.

Consider Feature Naming and Application Versioning Generally, when a customer licenses a new version of a vendor's application, they are entitled to continue running the current version of the application, provided that the total application usage of the new and old versions is within the appropriate license grant. Many vendors encourage this behavior by implementing FlexNet's date-based versioning, where the product license contains the rights to

application upgrades for the duration of the related maintenance agreement.

In such cases, it does not make sense for a vendor to change feature names across the different application versions.

However, vendors sometimes want to disable the use of prior features when distributing a new product version, usually as part of a product migration or obsolescence strategy. In these cases, vendors must change feature names across a related product set that contains similar functionality.

By following the feature naming convention of:

application_name.feature_type

vendors can better handle these situations because feature names are referenced under the name of the replacement application.

Utilize FlexNet Checkout Data to Provide Product Context
FlexNet can record detailed data about requested license features and store it in report logs for later reporting by vendors or end users. When you set checkout data you provide a link between a common component and the application that requested it.

A useful example of this is as follows:

In our example in diagram 1, the application Chart shares a common component with several applications. When Chart (chart.primary) calls the Print (common.print) component, FlexNet can log that the request for the Print component came from the application Chart (chart.primary).

By providing this data to FlexNet both you and your end-users will have the ability to understand how products are used and discover software usage trends.

Utilize FlexNet Checkout Data to Provide Product Versions

Our recommended best practice for reporting is that vendors pass the actual product version number (or version date) through to the license server as checkout data. This enables customers to report on the actual version of features used as well as the license version served up for that feature. Vendors typically encourage end users to move to the latest version of an application; however, unless vendors enable reporting on which versions are in use, there is no easy way to systemically identify users running old application versions.

Use FlexNet PACKAGES to Enable Easier Matching of License Files to Customer Purchase Orders

FlexNet provides two different ways (license keywords) to package products:

- **PACKAGE**. This is simply a means of bundling a set of licensed products or components in the license file, and has no impact on licensing or reporting behavior.
- **PACKAGE_SUITE_RESERVED**. This is a means of bundling a set of licensed products or components in the license file, and affecting the behavior of those licensed components at run time so that they are treated as one package bundle from a license usage perspective. As noted above, this option enables reporting of both the package use as well as the individual component use.

Accordingly, packages are useful for both simplifying the contents of a license file (PACKAGE) and for enforcing bundled license models (PACKAGE_SUITE_RESERVED).

The reason we recommend vendors use the FlexNet package capabilities is to make it easier for customers to match the license files they receive to the actual products they purchased. Consider the hypothetical product suite, Art Studio, containing the otherwise standalone products Paint, Draw and Sculpt.

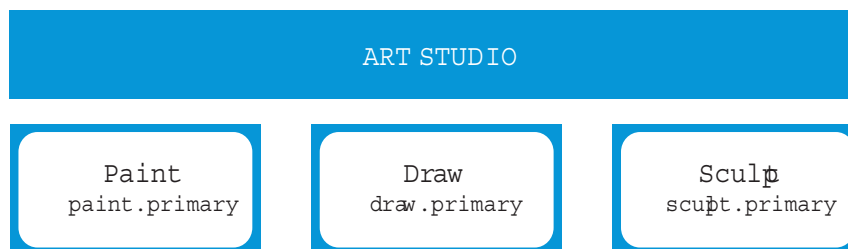


Diagram 2

Without the use of a FlexNet package, a customer that orders a copy of the Art Studio bundle to be installed on the host "Madison" would receive the following FlexNet license file (license file items have been abbreviated for clarity):

```
INCREMENT paint.primary vendor 1.0 permanent
uncounted \
HOSTID="Madison" . . . . .

INCREMENT draw.primary vendor 1.0 permanent
uncounted \
HOSTID="Madison" . . . . .

INCREMENT sculpt.primary vendor 1.0 permanent
uncounted \
HOSTID="Madison" . . . . .
```

From a purchasing perspective, this license file compares very unfavorably to the following license file:

```
PACKAGE ArtStudio vendor 1.0 COMPONENTS = \
"paint.primary draw.primary sculpt.primary"

INCREMENT ArtStudio vendor 1.0 permanent uncounted \
HOSTID="Madison" . . . . .
```

This second license file, while providing exactly the same license enforcement as the first license file, is substantially easier to match up against the item ordered and paid for.

FlexNet Certification Program

Flexera Software developed the FlexNet Certification Program to give vendors specific licensing implementation guidance, and to assist them in addressing issues faced by license administrators. The program also enables vendors and corporate end users to more easily deploy FLEX enabled applications in accordance with industry best practices.

Under the FlexNet Certification Program, Flexera Software certifies vendors' licensing implementations using criteria in the following areas:

- License distribution
- Product and license server installation
- Application behavior searching for licenses; handling license expirations; handling license server failure conditions
- Reporting - ability to accurately and clearly report license usage
- Warnings - availability and configurability of warning messages

Complete guidelines on the FlexNet Certification Program, including how to request an application for certification are available at www.flexerasoftware.com.

Summary

Many of today's licensing implementations fall short from an ease of use and reporting perspective. Without a defined licensing policy, vendors can end up with inconsistent licensing implementations across their product offerings. Furthermore, if a vendor's defined licensing policy is not customer focused, the vendor loses an opportunity to add significant product value for their customers. This situation commonly arises when the licensing toolkit is handed off to product development groups without the vendor initially determining its licensing policies and clearly communicating these to the various product groups.

The solution to these and other issues is to treat licensing as a tier-one product feature, instead of an afterthought. In addition, vendors should consider customer ease of use throughout the licensing implementation process.

The issue of improved licensing implementation is particularly time sensitive given the major initiatives in place at large end-user organizations such as Motorola, Chevron, Texaco, Shell, Honeywell, Pfizer, Northrop Grumman, AMD, Intel, Shell, BP, and NASA to implement global software licensing management and reporting infrastructures. With the push to centralize procurement and management, vendors with poor licensing implementations are finding it more difficult to hide, especially when selling to Fortune 500 companies.

Ultimately, improving licensing transparency is in the best interest of vendors themselves. Making it easy for customers to demonstrate that they are actively using your products, with increasing usage levels, also makes it easy for vendors to sell additional products and licenses. Clear and accurate usage reporting forms the basis for a robust set of licensing models under which customers can buy, and is a pre-requisite for the utility pricing models of today and tomorrow.

About Flexera Software Consulting

Flexera Software's consulting organization is available to help software publishers with FlexNet implementations. Our mission is to help software and embedded software vendors implement FlexNet in a timely and predictable way, reducing the time to benefit and increasing ROI. With over 20 years of experience with licensing technology and the FlexNet platform, Flexera Software offers a full range of consulting services including licensing assessment, strategy development, requirements development, onsite training, licensing and operations implementation, and FlexNet Certification. For more information visit www.flexerasoftware.com.

About Flexera Software

Flexera Software provides solutions that power the business of software for multiple customer segments, including hardware and software producers, engineers and developers, helping them uncover revenue opportunities, streamline their infrastructure and reduce costs. Flexera Software's proven solutions have been simplifying the business relationship between software and hardware producers and their enterprise and government customers for more than 20 years, enabling Flexera Software to maximize the value of the software the world develops and uses. For more information, please go to: www.flexerasoftware.com

About the Authors

Dan A. Griffith (Dan.Griffith@motorola.com)

Dan is manager of Motorola's Comprehensive Software Asset Management team (CSAM), which is responsible for distributing EDA software licenses to all Motorola design centers worldwide from central license servers in the US. CSAM has 20 license servers located in Arizona and Illinois. These servers logged 59 million hours of license usage and recorded over 499 million license events for 31 software vendors in 2002.

A 25 year Motorolan, Dan has 12 years experience in managing and supporting electronic licensing, as well as negotiating software licensing contracts for Motorola. Prior to the formation of CSAM in June of 1998, Dan managed the EDA software licenses and the UNIX compute environment for Motorola's ASIC Division. He also worked for Motorola's Government Electronic Group from 1978 to 1991 as a test manager.

Dan studied electrical engineering at the University of Toledo and holds a BS in Business Administration from the University of Phoenix. He lives in Scottsdale Arizona.

Richard A. Kline (rkline@flexerasoftware.com)

Richard is a strategy consultant in Flexera Software's Global Consulting Services organization. He is responsible for helping software companies plan and implement their electronic licensing strategies. He has more than eight years' experience in designing, implementing and maintaining electronic licensing and software delivery solutions. He has previously worked with more than 100 software companies, including Synopsys, Symantec, and Documentum.

Prior to joining Flexera Software, Richard worked at Intraware where he was responsible for Intraware's partnership with Flexera Software. During this time he designed FLEXIm for SubscribeNet, Intraware's hosted implementation of GTlicensing. Prior to joining Intraware Richard founded BITSource Inc. (which was acquired by Intraware in 1999) where he pioneered the concept of a digitally signed volume license. This process allowed software companies to eliminate their paper-based licensing, which is both costly and time consuming. Richard has also held product management roles at Synopsys and Apple Computer working on both software distribution and electronic licensing products.

Richard holds a BS in Information Systems from Drexel University in Philadelphia, PA.



Flexera Software, Inc.
1000 East Woodfield Road,
Suite 400
Schaumburg, IL 60173 USA

Schaumburg (Global Headquarters),
Santa Clara:
+1 800-809-5659

United Kingdom (Europe,
Middle East Headquarters):
+44 870-871-1111
+44 870-873-6300

Japan (Asia,
Pacific Headquarters):
+81 3-4360-8291

Australia:
+61 2-99-8-22-178

www.flexerasoftware.com