

DoD ESI White Paper

Software License Considerations for Third Party Software

Keeping pace with contractual considerations in an
evolving software marketplace

July 2013



About DoD ESI

The DoD ESI was formed in 1998 by Chief Information Officers at the DoD. To save time and money on commercial software, a joint team of experts was formed to consolidate requirements and negotiate with commercial software companies, resulting in a unified contracting and vendor management strategy across the entire department. Today, DoD ESI's mission extends across the entire commercial IT life-cycle to include IT hardware products and services. DoD ESI has established DoD-wide agreements for thousands of products and services.

www.esi.mil

Disclaimer

The content of this white paper is not provided as legal advice, but rather as general information designed to point out some of the business issues and considerations involving third party software. Readers should not rely on the content of this paper to make contract or other legal decisions. Drafting and negotiating contracts and software license agreements—including provisions addressing third party software—should be supported by legal counsel.

Table of Contents

Introduction and Background	4
What is Third Party Software	4
Defining Third Party Software	4
Software Types	5
Software Combinations	6
Third Party Software	6
Third Party Software Licensing Issues	7
When Licensed Separately from Third Party Publishers	7
When Licensing from One Publisher Under One License	8
Summary	10

Introduction and Background

Third party software is ubiquitous in the software industry. Very few software products work independently of other software products. Although some of the combinations have been around almost from the beginning of the industry, the nature and importance of these combinations are changing. Important concepts—including open source, embedded software, freeware, shareware, encapsulation, groupware, complementary software and others—have been added to our vocabulary.

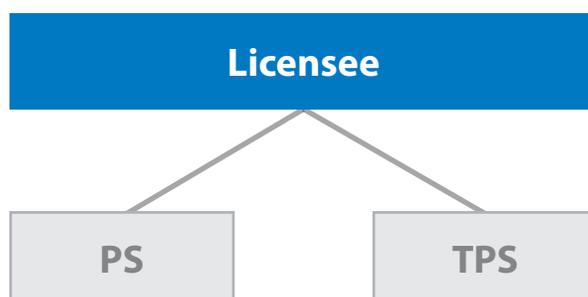
Many writers and industry analysts have rightly focused on the benefits provided by the proliferation of interoperable software while relatively few have discussed the less interesting, yet equally important, licensing challenges these combinations present. This paper provides education and insight about software combinations often encountered when negotiating commercial software licenses—and discusses options for dealing with them appropriately in the license terms and conditions.

What is Third Party Software?

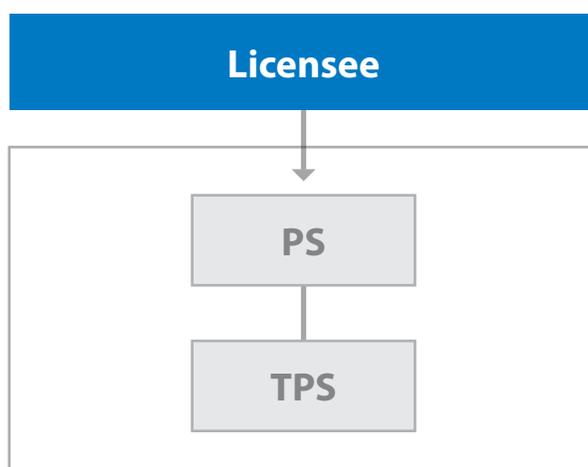
Defining Third Party Software

In its simplest terms, whenever a licensee is licensing one software product (let's call it primary software, or "PS") from the publisher or its reseller, any software created by other publishers or independent developers is considered third party software ("TPS"). There are two important examples where TPS creates potential licensing issues:

1. the licensee needs both PS and TPS to work together for some reason, and obtains PS in a license with the PS publisher and TPS in a separate license with the TPS owner/developer,



2. TPS is "bundled" with PS by the PS publisher, as part of a comprehensive solution, and is explicitly or implicitly included in the license between licensee and the PS publisher.



An Automobile Analogy. One analogy to this third party software scenario is found in the auto industry. Automobiles almost always include components made by firms other than the car manufacturer. Sometimes those components are incorporated in the vehicle at the car manufacturers' factory; other times the components are added by the dealer or by the consumer.

The auto example corresponding to the first software example above is when components are installed by the dealer or are purchased separately by the consumer from the individual manufacturers of specialty parts (or their retail distributors). Examples include replacing factory speakers, shock absorbers or exhaust systems to achieve a particular sound or look, or adding special lights, stand-alone navigation devices, seat covers, spoilers or even engine components. In some cases, these components are specifically designed to work with a certain make or model of automobile. In other cases they are potentially applicable to almost all cars. It is generally up to the consumer to make sure the separately purchased components are compatible with the car.

The automotive analogy to the second software example is when products from different manufacturers are bundled and sold by the car manufacturer as part of the car. The consumer buys all the bundled products together in one transaction. Tires, sound systems, transmissions, navigation systems, and a host of other car components are produced by other companies, but are sold as part of the car. The consumer has an expectation that all components will work together—and looks to the car manufacturer (through the dealer) to fix any problems, regardless of who manufactured the individual components.

Software Types

Let us examine TPS further by looking at some details about how and why PS and TPS work together, before discussing the licensing issues associated with them.

The software industry has developed a multitude of software types and categories. While there are many ways to categorize software, some find it useful to think of software in terms of its primary function, as follows:

- operating systems to execute commands;
- utilities to organize and maintain files or programs;
- applications for business transactions;
- databases to load, store, and retrieve data;
- middleware to connect applications;
- desktop apps for word processing, data analysis via spreadsheet, etc.

Software Combinations

There are two types of software combinations:

1. **Dependent Software Combinations.** One software product might require one or more other software products in order for it to perform its primary functions. We might say the ability of the first product to function is **dependent** on the other(s). Usually these dependency relationships are between different types of software.

Perhaps the most common example of dependency is the use of operating system software to execute commands between applications and the machines on which they run. This combination allows the application to use the machine's capabilities. Another common example is when application software uses database software to load, store, and retrieve information generated by the application.

In these examples, the interacting software elements are different types and are separate products, each performing its primary functions in combination with the others—for example, operating systems with applications, or applications with databases. In many cases these products come from different publishers. In some instances, publishers create software in two or more categories designed to work together “out of the box.” The first example (*software supplied by different publishers*) involves third party software, while the second one does not (*software supplied by the same publisher*).

(EDITOR'S NOTE: One example of licensing complications arises when a publisher uses—does not create, but uses—software from another publisher (i.e., third party software) in its product, and makes the products work together out of the box. We will address more on this scenario later.

2. **Complementary Software Combinations.**

In some cases, the interacting software elements come from the same functional category. For example, two or more applications might work together on different aspects of business transactions, while sharing a common operating system and a common database. These combinations of applications are usually thought of as **complementary** rather than dependent. Although this “same-type-of-software” combination is the most common complementary scenario, some complementary combinations involve different types of software.

Third Party Software

Depending on one's perspective, any of those software elements could be viewed by a software publisher—or more important, by a licensee—as TPS. The two basic requirements for labeling any software as TPS are:

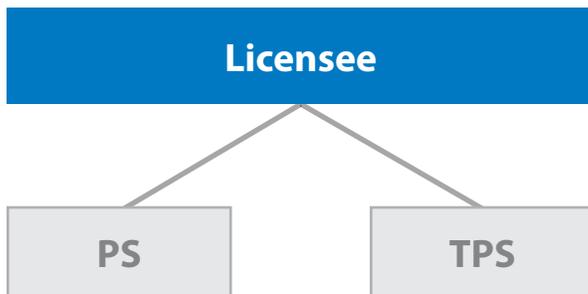
1. at least two software products working together; and
2. different publishers or developers separately creating or owning the intellectual property rights to each product, or (*in the case of open source software*) separately creating and making available the source code.

While there are many other examples of complementary or dependent software, the focus of this paper will deal with the two scenarios described in the definition section above:

1. PS and TPS are licensed separately from their respective publishers; or,
2. PS is licensed from the PS publisher where TPS is knowingly or unknowingly included with the PS.

Third Party Software Licensing Issues

When Licensed Separately from Third Party Publishers



As discussed above, primary software (PS) and third party software (TPS) are often licensed from their respective publishers separately. This can occur whether the PS is dependent on the TPS or is complementary to the TPS. The dependent software examples already cited include PS applications dependent on TPS operating systems, and PS applications dependent on TPS databases.

An example of complementary application software might be an Enterprise Resource Planning (ERP) system from one publisher providing customer order information, and other data, to a shop floor scheduling system created by another publisher—with middleware from a third publisher connecting the two systems.

When separately licensing multiple dependent or complementary software products from each publisher, there are two key license agreement considerations:

1. **Compatibility/Interoperability Warranty.**

Arguably, the most important licensing issue in the scenarios involving separately licensed PS and TPS is for the licensee to understand and document the extent to which each publisher guarantees compatibility or interoperability of the separate products. Obtaining these guarantees can be somewhat challenging, but they are necessary for meeting licensee's expectations for satisfactory performance of both PS and TPS.

The best acquisition practice is to include contractual language in all relevant licenses that states the degree of compatibility or interoperability among the licensed products. (*Strive for 100% compatibility or interoperability, if the publishers make that claim*). In order to give teeth to these promises, the licenses should specify reasonable remedies to protect the licensee in the event the products do not work together as promised—including a reasonable warranty period and the right to return the product for a full refund. If possible, include the right to be reimbursed for lost time and other costs associated with warranty breaches.

Generally, it is easier to obtain these “compatibility warranties” when dealing with operating systems or databases. The products that are designed or tested to work together are usually well known and advertised. That does not, however, remove the importance of getting the appropriate guarantees in the license.

Where it often becomes more difficult to obtain interoperability guarantees for complementary software products is when design and testing of the products working together have not yet occurred. In such cases, each publisher is likely to leave the risk of interoperability to the licensee. The best course is to be as specific as possible in the licenses about the extent of interoperability available, if any, and to write clauses holding the publishers accountable for that interoperability. Absent specific promises, licensees assume the risk of incompatibility.

2. **Maintenance & Support Obligations.** Another important factor in negotiating separate licenses for PS and TPS is to minimize finger pointing between the publishers when something does not work properly with one or both products. When you report a software issue, each publisher might claim its product is working as designed and might suggest the problem lies with the other product. Although there is no sure-fire contract solution to this problem, it accentuates the need for very clear definitions of warranted performance, as well as clear lines of product-support responsibility and responsiveness standards for reported problems, in each license.

2. the proliferation of open source software—it is readily available and it is cheap. (*NOTE: For more information about open source software, see the ESI white paper, “Considerations for Open Source Software Use.”*¹⁾)

The resulting PS license implications are significant. The following six contractual considerations address intellectual property (IP) concerns as well as financial, software performance, and other safeguards.

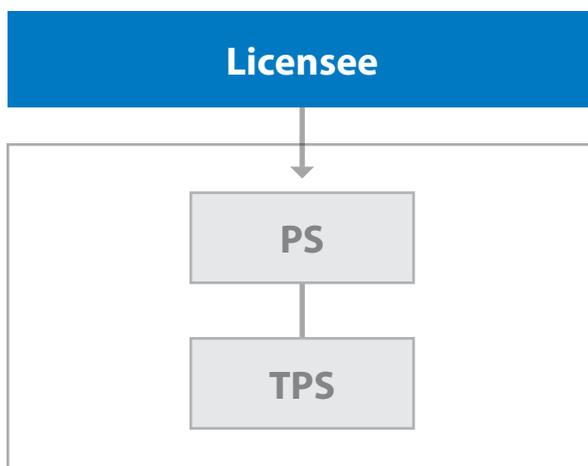
These considerations apply whether the TPS is open source or proprietary. All software, including open source, carries potential license obligations that could be problematic to licensees obtaining PS with TPS included. All licenses should address these concepts regardless of whether the presence of TPS is known or not.

Please note that, to be effective, all of the following items must be included in the license agreement:

1. **Disclosure of TPS.** (*A statement of assurance by the PS publisher that all TPS included in, or with, the PS is fully disclosed in the license agreement. Disclosure should be by way of an attribution list—a list of TPS with the third party publishers’ names and copyrights (or copylefts, for open source).*) This requirement is fairly simple and straightforward, although publishers might push back for various reasons—they might not know about all TPS or they might see disclosure as unnecessary or inconvenient. Don’t be pushed into ignoring this license requirement. Demand to see the PS publisher open source compliance policy or the results of open source code scans. The importance of full disclosure is self-evident—it is mandatory for licensees to know the presence and source of all open source code or proprietary IP included in or with the PS. Licensees should know what they are receiving and the potential obligations that come with it.

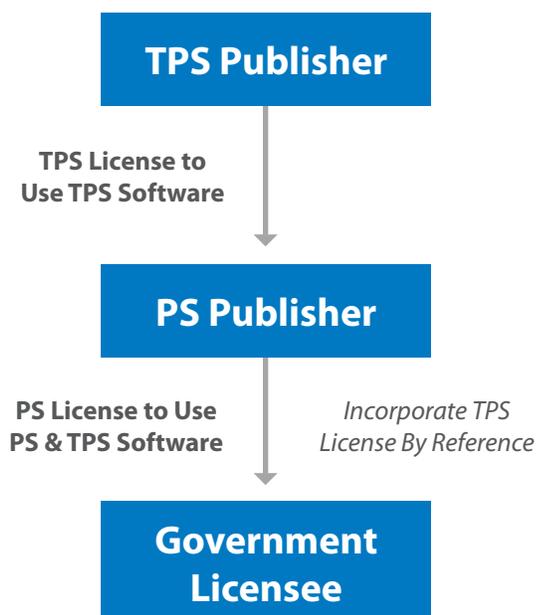
When Licensed Separately from Third Party Publishers

The more interesting and rapidly growing TPS scenario is when the PS publisher incorporates, embeds, or includes TPS with its PS under a single license. The examples in today’s marketplace are numerous and varied. More than ever, publishers are using TPS directly in their applications. This stems from two separate, yet related, phenomena:



1. most software publishers have accepted the fact that they can’t be the sole source of software solutions and should leverage other software where it makes sense; and

2. **Publisher’s Right to Use the TPS.** *(A covenant from the PS publisher that it has the right and authority to include the TPS with the PS in the manner in which that TPS is included.)* Publishers who refuse to disclose TPS, or to covenant that they have the right to include it or use it with their PS, should be viewed as skeptically as the seller of a car who can’t produce the title. In the case of software, publishers are not normally selling licenses with IP from other publishers fraudulently. They could, however, be doing so mistakenly or unknowingly—and the results could be costly to the licensee. It is imperative that licensees receive full disclosure and assurance of the PS publisher’s right to use or include the TPS by way of a covenant from the PS publisher to licensee. Also, these rights to use or distribute TPS should exist in licenses between the PS publishers and TPS publishers, so the covenant from the PS publisher to the PS licensee should refer to those TPS licenses. Although the covenant from the PS publisher to the licensee is probably sufficient protection, in some cases licensees might demand to see the TPS licenses with the PS publisher *(or to incorporate them by reference into the PS license)* to ensure the rights are adequately specified.



3. **PS Publisher Indemnification of TPS IP.** *(A covenant from PS publisher that it has appropriate Intellectual Property (IP) indemnifications in its licenses with TPS publishers—and indemnification to the licensee from the PS publisher not only for its own PS IP, but also for potential IP infringements involving the TPS.)* While acknowledging that this topic could be fraught with complicated legal issues, the basic idea here is to ensure that the licensee not only has assurances from the PS publisher regarding that publisher’s authority to use or include the TPS, but also that the PS publisher agrees to protect the licensee from any claims of infringement of the TPS IP by any party. Such claims could come from the TPS publisher or from some other party claiming ownership of the TPS IP, and the charges could be made against the TPS publisher, the PS publisher, or the licensee *(by virtue of the licensee using the TPS)*. An indemnification from the TPS publisher to the PS publisher in their license is likely ineffective for the PS licensee due to privity issues regarding the contract *(i.e. there is no privity between the PS licensee and TPS publisher)*. Since “open source,” by definition, is not IP, the open source license obligations regarding appropriate use also should be included in this clause.

4. **No Additional Licenses or Fees.** *(A covenant and indemnification from the PS publisher that the licensee will not be required to enter into additional licenses for TPS, or owe additional fees of any kind to any third party, as a result of licensing PS from the PS publisher.)* The licensee needs to make sure there are no surprises in the form of unexpected licenses or fees—even for open source code. PS publishers might not be aware that TPS *(especially open source TPS)* has been used in the product, or they might think they are complying with TPS license requirements, but are not. Therefore, the PS publisher is unable to promise unequivocally on behalf of known or unknown TPS publishers that there will be no additional licenses or fees. As a result, this covenant needs

to include an indemnification by the PS publisher to protect the licensee fully, meaning that the PS publisher will cover any unexpected licenses or fees—financially and otherwise. Although it seems common sense and basic, many licenses for PS that include TPS are silent on this matter. The potential liability could be significant.

5. **Product Warranty Includes TPS.** *(A covenant that the PS publisher's product performance warranty includes and extends its coverage to the TPS provided with the PS.)* When publishers undertake to combine TPS with PS, they should be held accountable for the performance of all the software—including the TPS—and those promises should be explicitly covered by the product performance warranty from the PS publisher to the licensee. This should be the case even where the PS publisher encapsulates the TPS (*wraps it outside of, or separate from, the PS IP*) to insulate it from claims of enhancement or IP infringement. Those IP protections afforded by encapsulation of TPS do not apply to its performance as part of the combined PS and TPS solution. As with other aspects discussed above, promises made by TPS publishers to PS publishers in their licenses do not protect licensees of PS due to a lack of privity between the PS licensees and TPS publishers.
6. **No Obligation to Disclose or Share TPS Enhancements.** *(A covenant from the PS publisher that the licensee will not be required to disclose or share enhancements to open source or proprietary TPS software—whether made by the PS publisher during PS development or implementation or by the licensee, knowingly or unknowingly, by virtue of implementing or using the TPS.)* This concept is rooted in a requirement, found in some open source licenses, for software users to share enhancements with the open source community. It is advisable to include a clause in the PS license that relieves the licensee of any obligations to share enhancements of open

source or derivative works of proprietary TPS. As stated above, encapsulation is one mechanism used by publishers to avoid the open source enhancement issue or to circumvent ownership issues regarding derivative works of proprietary TPS. Again, although this topic can involve a very complicated set of legal issues, the key idea is to make sure the licensee is protected against unwanted disclosures or sharing of improvements or enhancements, especially where the licensee's confidential or proprietary information is involved in those enhancements.

Summary

Third Party Software is now used by many, perhaps most, software Publishers. Sometimes the third party software is necessary to the effective use of the primary software being licensed. In other cases it is optional or convenient to combine products. When these products are licensed separately, the key objective is to document accurately, in each license, the degree of interoperability among the products and the support obligations of each publisher.

Many PS publishers use open source as part of their products; some incorporate proprietary software from other publishers. In either case, when the PS publisher then licenses its software with the third party software included, it creates a special set of legal challenges. Those issues require certain clauses in PS licenses, starting with full disclosure of all third party software in the PS, the right to use TPS, performance warranties, and other important protections and indemnifications.

About the Author

Tom Crawford is a seasoned executive and consultant who has been leading and advising technology businesses for the past 20+ years. After successful senior management roles with leading software companies— including SAP, PeopleSoft, Oracle, and BMC— Tom started his own technology consulting business, lending his expertise to several clients in various industries. Tom currently serves as a software contract and procurement subject matter expert for the BuySide Partners team supporting DoD ESI. Tom’s experience leading business units and sales teams that have negotiated and closed scores of software and services contracts ranging up to \$65 million provides the foundation for his expertise and advice relating to software industry sales practices, contract negotiation strategies and commercial best practices for software license terms and conditions. Tom is a graduate of the U.S. Naval Academy, a former U.S. Navy officer, and holds an MBA from Wharton and a Juris Doctor from the University of Pittsburgh. His work with DoD ESI draws upon his diverse experience helping clients save significant dollars in software and services procurement.

References

¹ Considerations for Open Source Software Use, http://www.esi.mil/Uploads/DoD-ESI_WhitePaper_Open_Source.pdf



DoD ESI is an official
Department of Defense initiative
sponsored by the Department of Defense
Chief Information Officer (DoD CIO).

**Your Preferred Source for
IT Acquisition Across the DoD**

- BEST VALUE**
- EFFICIENT**
- LOW RISK**
- VOLUME DISCOUNTS**
- UNIFIED VOICE**

Visit DoD ESI online at www.esi.mil

Department of Defense Chief Information Officer
6000 Pentagon
Washington, DC 20350-6000