Software Escrows as Part of an Intellectual Property Strategy

BY RAJIV PATEL

Introduction

For many companies, a key aspect of a comprehensive intellectual property strategy is to identify and enforce mechanisms to protect their investment in software purchased from software developers. One particular tool to guard such investments is the software escrow.

What is a software escrow?

A software escrow is a deposit of source code of software and other materials with a third party escrow agent. Generally, a party licensing the software (the "licensee" or "buyer") requests the software escrow from the owner of the software (the "software developer" or "developer") to ensure maintenance of the software and possibly performance of development obligations under a license.

Why does a software escrow exist?

Software developers typically make a significant portion of their profit on recurring maintenance contracts instead of the basic license fees. Often, software developers will even forego basic license fees and focus instead on ensuring that the licensee is captive for longer periods through the use of maintenance contract, which may include services in addition to maintenance of the code itself. Thus, most developers offer software licenses that only license object code, i.e., the code that can be read by a machine, rather than the source code, i.e., code that can be deciphered and read by a person. Aside from leveraging maintenance contracts to develop longer term relationships with a licensee, software developers also have a vested interest in protecting the source code from risks that directly affect the source code. Examples of such risk include copying or reverse engineering the code to develop a competing product or unauthorized modifications to the source code that may affect performance or operation of other parts of the code. On the flip side, licensees reliant on such software developers want to ensure that their investment in the developer's software is protected and not lost if the developer fails to fix bugs or the like. Hence, such licensees want may need access to the source code in the event that the software developer no longer provides the object code.

In what situations are software escrows requested?

Often, there is tension between the software developer's desire to keep source code confidential and out of the hands of the licensee and others who may gain possession of, or knowledge about, the source code, and the licensee's desire to have access to the source code in the event that the software is not longer available at agreed upon levels of service.

Examples of when a licensees generally request a software escrow include an established company integrating software from a small, relatively unproven company into its product or service offerings or a business integrating a developer's software into its business such that the business could be halted if the software were suddenly unavailable or did not perform to expectation.

Who is likely to agree to a software escrow?

Practically, smaller software developers are more likely to agree to a software escrow than larger software developers. Smaller software developers want to gain the trust and confidence of larger companies in doing business with them. Moreover, they seek to do this without exposing their source code and associated intellectual property to the large company, for obvious business reasons of having the endorsement of a larger client to attract other customers or for having their products bundled with those of larger developers to go to an even larger end-user base.

What types of risks does a licensee seek to minimize by a software escrow?

Generally, are three types of risks that drive creation of a software escrow. The first risk is that the software developer substantially goes out of business or becomes financially unable to perform its development and support obligations. The software developer does not yet file for bankruptcy and still exists as an entity, but fails to provide support or improvements to the software.

The second risk is that the software developer files for bankruptcy and terminates the license in the case. Under the U.S. Bankruptcy Act, if reorganization occurs in a Chapter 11 bankruptcy the software developer is still in business, and the software developer elects to continue performing its license obligations, there may be no significant impact on the licensee of the Chapter 11. If, however, the software developer ceases operations or otherwise fails to provide support at a previously agreed upon level with the licensee, licensee will lose a vital part of its strategy regarding the license software and, absent a triggered source code escrow, may find itself unable to maintain and develop its products.

The third risk is that the software developer is acquired by a competitor of the licensee. In such situations, the software may be altered to the detriment of the licensee, discontinued, or support for it may be dropped altogether. An associated risk is a change in the relationship between the software developer and the acquiring third-party. For example, the acquiring third-party fails to provide support at a level agreed upon with the software developer.

In summary, the risks above highlight for the licensee the issue that business failure, however it occurs, creates the problem that you either have to have the source code deposits and rights to use them under the escrow or resort to more expensive and less reliable approaches such as replacing it with another software component and the consequent reengineering costs, or buying a new license from anyone who comes into possession of the software through acquisition without the support obligations under the old licenses. Moreover, access to the source code and related remedies depend on licensee's ability to replace the software developer's services, for example, either through doing it for itself or getting a third party to replace the software developer.

What type of risks does the software developer seek to minimize?

There are two primary risks that the software developer seeks to minimize through source code escrows. The first is minimizing the risk of losing business. The software developer can use the escrow to remove uncertainty in business dealings between the licensee and them so that the licensee feels comfortable in entering into a deal with the software developer.

The second risk to minimize is the risk of releasing the source code from escrow. As a part of this risk assessment, the software developer will need to maintain flexibility on circumstances in which the source code would not be released from escrow due to appropriate business decisions. For example, if a software product reaches end of life, the escrow agreement should be structured so that it does not trigger a release event, particularly, if there is an available migration or upgrade path available to the licensee.

How does a software escrow work?

Thanks to well established software escrow practices, it is relatively easy to find a base framework creating a commercial software escrow particular for the needs of a deal between two parties. Below are some basis considerations on what such frameworks include and where issues may arise.

Identify a software escrow agent.
Initially, the software developer and the software developer must agree upon a software escrow agent. Key in determining who to select as a software escrow agent is that they are an independent third party unaffiliated with the software developer and that they have experience in administering source code escrows. Examples of established software escrow agents include Iron Mountain Intellectual Property Management, Inc., (www.ironmountain.com), SourceHarbor, Inc. (www.sourceharbor.com), and EscrowTech International, Inc. (www.escrowtech.com).

Negotiate what goes into the software escrow.
The licensee should ensure that the complete source code of the licensed software goes into the escrow, along with associated materials such as documentation, software libraries, appropriate third-party items, and the like. The deposited source code should preferably be in electronic format. In addition, the licensee should consider whether the software developer should also include documentation such as development manuals and the like. In any event, the rights to have possession and use of the deposit materials, whatever they are, should depend on the agreement to deposit them, and not on actual deposit.

Negotiate how often updates go into the software escrow.
The licensee should ensure that the software developer is obligated to update its source code deposits with all-new versions, updates, and new releases of the licensed software. Moreover, the obligation for these additional deposits should be satisfied within a short time period after

the initial distribution of the new version, update, or release. Such time periods are typically within 30 days or less.

Confirm what went into the software escrow.

A key part of the escrow arrangement is ensuring that what goes into the escrow is what the licensee expects. Deposits of incomplete or out-of-date source code do happen, but can be prevented. To avoid such problems, include provisions in the escrow agreement for deposit of new versions, updates and releases as previously described, in addition to the original deposit of code.

Determine release conditions from the software escrow.

A critical aspect of the software escrow is the release provision. The release provision, typically referred to as the release conditions, defines the conditions upon which the source code and associated materials are released by the escrow agent to the licensee As a general matter, release provisions should focus on identifiable, indisputable facts. The easier it is to demonstrate the occurrence of a release condition, the more quickly and cheaply the licensee should be able to get access to the deposit materials. In addition, release conditions must also legally enforceable, an important factor in the context of bankruptcy proceedings (further described below).

Many source code escrows limit the release event to the situation in which the software developer has ceased doing business and, therefore, cannot maintain the software. In this regard, licensee can also ask for conditions related to harbingers of impending failure for release: the appointment of a receiver for software developer's business, the making of a general assignment for the benefit of creditors (an alternative to bankruptcy liquidation), and the announcement to the public in general that software developer is ceasing operations, are examples of such release conditions. Still another release consideration to plan for is a software developer's outright refusal to perform, repudiation of the license, including rejection in bankruptcy.

A licensee may also consider attempting to procure additional release events tied to the software developer's failure to perform its maintenance obligations in a timely or effective manner, e.g., a consistent failure to respond to, or correct, documented errors within a specified number of days of their report by the licensee to the software developer.

The licensee should also consider other release events tied to the practicality of using the software developer for continuing maintenance services. For example, the licensee may use this release when the software developer seeks to increase maintenance costs beyond a specified limit set in the maintenance contract.

Next, avoid relying on "insolvency" as a release condition because its occurrence is difficult to determine with accuracy and it is usually determined only in hindsight through expensive litigation with delay. Finally, note that though the release condition is common, software developer's filing for bankruptcy is not an enforceable release condition.

Determine who pays for the software escrow.

Typically, the software developer will pay for the software escrow, although this point is often negotiable between the parties.

What are key issues faced by the licensee with respect to a software escrow?

The material (i.e., the source code itself) is never deposited, or only partially deposited, into the escrow.

The licensee should also watch for and monitor the escrow account activity, which should include a description of what was deposited and when. In addition, consider a technical verification service offered by independent third parties, including some software escrow agents. These services verify the integrity of the deposited materials and can range from simple tests that confirm the physical content of the media to actual compilation of the code to test functionally. The verification service can be helpful in ensuring that the licensee receives complete and useful source code when the release event occurs.

The occurrence of release conditions is unclear or disputed.

Most escrow agreements allow the software developer to oppose the licensee's release request when a release event allegedly occurs. To address such conflicts, there are options available to write into the release process of an escrow agreement to avoid litigation over source code. The most common escrow provision to resolve a release in dispute is arbitration. Most parties to an agreement prefer this provision because it provides a quick, low-cost, and more importantly, decisive alternative to litigation.

Another approach for disputed releases is to have decision making executives for each party meet and resolve their differences. In this context, the escrow agreement and the

threat of a release act as catalysts to facilitate action and dialogue between the two parties. Still another approach to resolve disputes is to write into the escrow agreement the requirement that disputes be "expedited," including setting forth timetables to complete each step.

More influential licensees may be in a position to negotiate a release-on- demand clause in their escrow agreements. This clause instructs the software escrow agent to release deposit materials immediately after receiving a request from the licensee.

The source code is not helpful in a vacuum.
Ensuring that escrowed source code is complete and useful does not guarantee that the licensee will be able to work with it. To address this issue, consider requesting supporting documentation and a list of technical maintenance personnel from the software developer that can be included as part of the deposit. If the software developer goes out of business, these employees may be available as consultants after the release has occurred, helping to ensure that the licensee has access to support personnel familiar with the software. Be sure they are authorized to assist in the event of a trigger, despite any confidentiality agreements with software developer??

If supporting documentation or a list of technical maintenance personnel is not provided, the licensee should be prepared to hire an outside consultant or dedicate internal personnel to maintain the technology. Here, the licensee's goal would be to use the released source code to have the consultant maintain it for as long as needed or until a suitable replacement technology is available. In some instances, the licensee may desire to incorporate a predetermined number of hours or costs that would be covered under the agreement as it relates to these consulting arrangements. Likewise, a software developer may desire to cap such hours or costs if they end up in the agreement.

The software developer is in bankruptcy.
Bankruptcy is often the most complicated issues in structuring and enforcing an escrow agreement. It is important to understand the underlying principles because the Bankruptcy Code interferes with the parties' ability to strike any bargain they want. Section 365 of the U.S. Bankruptcy Code gives debtor parties to "executory contracts" (those with substantial performance remaining on both sides at the time of filing) a choice of assuming and finishing the contract or "rejecting" it, generally leaving

the other party to a damages claim. If debtor is a software developer of certain types of intellectual property (and note the definition of intellectual property in Section 101 of the Bankruptcy Code does not include trademarks, foreign patents or foreign copyrights), and rejects the license, the Congress also allows the licensee to retain certain rights to use the technology as it existed on the date of filing in exchange for paying royalties due under the agreement, AND to have rights under "ancillary agreements," which includes the escrow agreement.

Section 365(e) generally disables certain contract clauses and provisions in some non-bankruptcy law that permit the non-debtor's termination of an agreement because of a debtor's bankruptcy filing or its financial condition upon filing. These so-called ipso facto clauses are not enforceable in licenses where the debtor is a software developer. Hence, special crafting is required for the escrow release triggers to be sure that if the license occurs, and the licensee makes the Section 365(n) election, the escrow release will occurred and be effective. Specifically, Section 365(n) grants the licensee absolute entitlement to retain rights to intellectual property (despite the debtor-software developer rejection of the license agreement) with some conditions. The licensee cannot also enforce exclusivity against the debtor or its assignee if the license is exclusive, but it cannot compel further performance (such as development or maintenance) otherwise provided under the license. Therefore, when drafting agreements for the U.S. the licensee should structure the licensing agreement to fall under the scope of 365(n).

There are a number of things a licensee can do to increase the probability of having the agreement fall under the scope of 365(n), including using executory contracts (which include software licensing agreements) that call for continuous performance over time by both sides. Because the rights that can be retained upon rejection are those which existed at the time of the bankruptcy petition, license grants must be present grants, not those which spring into existence upon bankruptcy. Specifically, escrow agreement should be drafted as a present grant to use the escrowed materials and not just a license right becoming effective upon bankruptcy. If the license makes the escrow agreement effective upon bankruptcy, then that does not fall under the section of the code that says the license must be in effect in order to be valid, e.g., reciting "software developer hereby grants" instead of "software developer grants."

The right to the deposit materials should also be without regard to whether the software developer actually made the deposit required under the license. That way, if the software developer did not do so and the escrow is triggered, the licensee will have a legal right to demand that the debtor or its trustee hand over the materials. There may be practical problems in such a situation, and escrows need to be monitored for compliance to avoid them, but at least licensee will have the legal rights to the materials and can work with the debtor or his trustee to find them.

Finally, the parties need to be careful in their license to differentiate which payments are for use of the licensed software and related intellectual property and which for maintenance. Licensees making the Section 365(n) election must pay royalties required under the license to continue using the intellectual property. If the license is not clear, the licensee will end up litigating how much "royalty" must be paid to continue its use rights under Section 365(n) without the support and maintenance of the software developer.

Conclusion

Software escrows can be a vital part of a software purchaser's intellectual property plan and strategy. The ability to access source code in the event of a software developer being unable to further maintain supplied software code can be critical for a licensee.

About the Author

Rajiv Patel (rpatel@fenwick.com) is a Partner in the Intellectual Property and Patent Groups of Fenwick & West LLP. Fenwick & West LLP is a high technology law firm based in Silicon Valley, California and is on the web at www.fenwick.com. Special thanks to Ralph Pais and Mark Porter of Fenwick & West on their insights and input for this article. Thanks also to Ric Sala of Iron Mountain Intellectual Property, Inc., Bea Wray of SourceHarbor, Inc., and Jon Christiansen of EscrowTech International, Inc. for their perspectives.