



The Accidental LICENSOR: Advanced Issues in Software Licensing

Reprinted with permission of the authors Johanna L. Werbach and Ron N. Dreben and copyright holders Moody's Investors Service, Morgan, Lewis & Bockius LLP, and the American Corporate Counsel Association/Global Corporate Counsel Association as it originally appeared: Johanna L. Werbach and Ron N. Dreben, "The Accidental Licensor: Advanced Issues in Software Licensing," *ACCA Docket* 21, no. 2 (February 2003): 54-71. Copyright © 2003 Moody's Investors Service, Morgan, Lewis & Bockius LLP, and the American Corporate Counsel Association/Global Corporate Counsel Association. All rights reserved. For more information or to join ACCA, call 202.293.4103, ext. 360, or visit www.acca.com.

Johanna L. Werbach and Ron N. Dreben, "The Accidental Licensor: Advanced Issues in Software Licensing," *ACCA Docket* 21, no. 2 (February 2003): 54-71.

BY JOHANNA L. WERBACH
AND
RON N. DREBEN

As more in-house information technology (“IT”) departments create or commission custom software applications, non-IT companies increasingly find themselves acting as software developers. Some companies have begun to license their internally built accounting, inventory, human resources (“HR”), supply chain, or other software systems in order to capitalize on these developments. Such companies may be tempted simply to create a licensor form for this purpose, based on a recent license-in transaction. Although this step would be a start, the “accidental” software licensor should be aware of additional, complex issues.

If you would like a primer or a refresher on software licensing, read the article by Karen L. Boudreau, “An Introduction to Software Licensing,” *ACCA Docket* 20, no. 9 (2002): 54–71, available on ACCA OnlineSM at www.acca.com/protected/pubs/docket/on02/software1.php. This article follows on that primer and asks you to consider likely questions that would raise the following advanced software licensing issues that you may encounter in the event that your company becomes a commercial software developer: marketing software that contains third-party source code, sharing rights under government contracts, reverse-engineering software, using a clean room, handling encryption and export issues, protecting your intellectual property, and dealing with a vendor’s bankruptcy.



Johanna L. Werbach is assistant general counsel, intellectual property and technology, Moody's Investors Service, in New York. She is available at johanna.werbach@moody.com.



Ron N. Dreben is a partner in the Intellectual Property Group at Morgan, Lewis & Bockius LLP in Washington, DC. He is available at rdreben@morganlewis.com.

MARKETING SOFTWARE THAT CONTAINS THIRD-PARTY SOURCE CODE

Q: Suppose that your head IT developer has a new accounting software package that he believes will be useful enough to license to third-party businesses. “We saved a lot of time by using free source code from the internet,” he tells you, proudly. “There’s no problem with that approach, right?”

A: It depends on the type of source code that your developer used. You may have a copyright infringement problem.

To assess this situation, you’ll need to determine whether the source code was shareware, freeware, or open source software (as discussed below). None of this software is in the public domain—that is, it is all subject to copyright protection—but the grant of rights under open source licenses is typically far broader than the rights accorded by a shareware or freeware license because those licenses can be as restrictive as licenses associated with proprietary software, such as Microsoft® Word.

Source Code Licenses

Shareware is usually free of initial cost, but with a license limited in either time or scope. A shareware licensee is given temporary use of a piece of software or unlimited use of only selected features

of the software at either low or no cost in order to entice the licensee to sample it. The shareware licensor generally hopes that the licensee will decide to upgrade to a full license for a fee. With “freeware,” the licensee receives a royalty-free license to use the software, usually with no set term. Under either of these licenses, creation of derivative works may be restricted, and you cannot infer a derivative right from silence. Indeed, in most cases, shareware/freeware licenses do not provide access to the underlying source code.

In contrast, open source software source code is made public for use or modification, freely subject to a very open-ended license. According to the Open Source Initiative (“OSI”)¹, an open source license² (1) imposes no royalty or other fee upon redistribution by a licensee (such as your company), (2) provides easy access to the source code and permits its redistribution, (3) allows modifications and derived works to be made of the licensed software and distributed under the same terms as the original software license,

THE SHAREWARE LICENSOR GENERALLY HOPES THAT THE LICENSEE WILL DECIDE TO UPGRADE TO A FULL LICENSE FOR A FEE. WITH “FREWARE,” THE LICENSEE RECEIVES A ROYALTY-FREE LICENSE TO USE THE SOFTWARE, USUALLY WITH NO SET TERM.

(4) cannot discriminate against selected persons, groups, or fields of endeavor, and (5) conveys rights that cannot depend upon the program being part of a particular software distribution. Open source licensees are encouraged to create and confer derivative works. The licensors’ proprietary rights are deemed secondary to the licensees’ rights to use the software, while freeware/shareware are still very much proprietary. Subject to certain conditions, an open source licensee may use OSI’s certification mark. OSI identifies many such model licenses on its website, including the Apple Public Software License and the Mozilla Public License.³

General Public License

Suppose that your programmer tells you that he has used GPL code in his derivative software. What does this code mean in terms of your company's rights? What type of license do you have?

The Free Software Foundation ("FSF")⁴ offers free software pursuant to its general public license ("GPL"). The FSF requires licensors to grant four basic freedoms to end users in order for their software to qualify for a free software license. These freedoms are similar but not identical to the open source license requirements.⁵

If you wish to redistribute GPL-licensed software in original form, modified form, or incorporated wholly or in part in another piece of software, you must observe the GPL license terms. This rule means that you must take care to distinguish new, original software that operates with GPL-licensed software from the GPL-licensed software itself because the latter must be shared under a license at least as broad as the GPL. In other words, by integrating a modified version of GPL-licensed software into software that your company would otherwise want to maintain as proprietary, you must make the modified source code available to end users. Your company would retain copyright in its derivative work, but your distribution of that work would be subject to the GPL's extremely broad grant of rights to third parties—that is, public sharing of the source code.

The FSF distinguishes between "mere aggregation" of GPL-licensed code with a user's proprietary software and "combining two modules into one program."⁶ If your company were simply to put two programs side by side on a CD-ROM or hard disk, your proprietary software would not be subject to the GPL. If you were to combine two pieces of software so that one did not operate without the other, the entire derivative might be subject to the GPL's broad grant.⁷ If Microsoft® had used a GPL-licensed web browser in Windows®, someone would have argued that Windows had become open source!

The GPL also disclaims all warranties to its users. Accordingly, your internal developers would need to test thoroughly any open source code that it uses for performance, security, scalability, or any other relevant factors. You have no place to turn to if the software disappoints. Further, if any purported open source code infringes a third-party right, your company may unknowingly become an infringer, as well.

For these reasons, the license of the open source code that you use in your company's software significantly affects the scope of protection that you can assert against third parties for unauthorized use of your product. As a practical matter, it will also affect any copyright application that you may prepare for your company's product.

Practice Tip

If you use GPL-licensed software internally only, the downside of using free software is generally limited, but you may encounter significant legal and business drawbacks to using it in applications that you commercialize and license to third parties.

SHARING RIGHTS UNDER GOVERNMENT CONTRACTS

Q: Your procurement department has worked with IT to develop supply chain software that has proved very useful, and the company has decided to commercialize it. The U.S. Department of Defense ("DOD") has expressed interest in it and has asked your company to significantly modify the software so that it is compatible with DOD's existing system. "Are there any legal problems with our going forward?" your procurement manager asks you.

A: A definite maybe. You should be concerned about the U.S. government obtaining rights in your software.

Government Rights

The federal government can obtain three different levels of rights in computer software: unlimited rights, government purpose rights ("GPR"), and restricted rights. What the government gets largely depends on the funding that it contributes to development of the software. Generally, the government obtains unlimited rights in software developed exclusively with government funds, meaning that all development costs were charged directly to a government contract, GPR in software developed with mixed funding, and restricted rights in software developed exclusively with private funds. These issues are addressed in the Defense Federal Acquisition Regulation Supplement ("DFARS"), which implements and supplements the Federal Acquisition Regulations as they pertain to DOD.⁸

The government also may obtain unlimited rights in publicly available (commercial) software, in software obtained with unlimited rights under another, but not the main, government contract or as a result of negotiations, and in software furnished with GPR or restricted rights if the restrictive conditions have expired. Additionally, if you furnish software to the government without restrictive markings, the law presumes it to have been delivered with unlimited rights.⁹ With unlimited rights, the government can use and modify the software (if source code has been provided) for any purpose and allow others, including nongovernmental entities, to use and modify the software for any purpose.

The government obtains GPR when software is developed partially with costs charged to indirect cost pools and/or costs not allocated to a government contract and partially with costs charged directly to a government contract. If your company were to modify its software, according to a DOD request, you would probably have a mixed funding situation. GPR lasts for five years, unless otherwise negotiated. During this period, the government has the right to use the software internally without restriction. It also may release the software to a nongovernmental entity and authorize the recipient to use the software for a governmental purpose. The recipient, however, must be subject to a use and nondisclosure agreement or be a government contractor performing a contract that includes the DFARS "Limitations on Use or Disclosure of Government Furnished Information Marked with Restrictive Legends" clause¹⁰ and that allows use of the software only when performing an activity in which the government is a party. "The contractor shall use technical data or computer software received from the Government with [GPR] legends for government purposes only. The Contractor shall not, without the express written permission of the party whose name appears in the restrictive legend use, modify, reproduce, release, perform, or display such data or software for any commercial purpose or disclose such data or software to a person other than its subcontractors, suppliers, or prospective subcontractors or suppliers, who require the data or software to submit offers for, or perform, contracts under this contract. Prior to disclosing the data or software, the contractor shall require the persons to whom disclosure will be made to complete and sign

the non-disclosure agreement. . . ." "Government purpose" includes competitive procurement, but "does not include the right to use, modify, reproduce, release, perform, display, or disclose . . . the software for commercial purposes."¹¹ After the GPR period expires, the government obtains unlimited rights in the software.

The government obtains restricted rights in non-commercial software developed exclusively at private expense. Restricted rights permit the government to use the software on one computer at one time (unless otherwise negotiated), to modify the software and use the modifications in a restricted manner, to make the necessary number of backup copies of the software, and to transfer the program to another government agency as long as the transferor destroys all copies of the software and notifies the party asserting the restricted rights of the transfer. The government also can permit contractors performing related service contracts to use the software to diagnose and correct deficiencies in a program and contractors performing emergency repairs to use the software. The contractor must be subject to a use and nondisclosure agreement or the "Limitations on Use" clause. It cannot reverse-engineer the software. (See "Reverse-Engineering Software" below.)

Although in none of the above circumstances would your company lose your copyright entirely to the government, you would probably find it more difficult to license your product to other government agencies or even to a private entity if the potential customer could get the same thing from the government. Further, in the case of unlimited rights and GPRs after five years, the government can publish or transfer your source code to third parties, thus making the software much less valuable.

Practice Tip

If you wish to limit the government's rights in software, you must place restrictive markings on the software.¹² These markings almost always take the form of authorized legends¹³ and must be conspicuous and legible.¹⁴ You should place the markings on the transmittal document or software storage container and on each page of accompanying written material.¹⁵ Because in some circumstances an apparently private sector customer, such as a university or a foundation, may actually be using govern-

ment funds to pay for developing or licensing software, you should include the markings in every software license and development agreement to avoid losing valuable intellectual property rights.

REVERSE-ENGINEERING SOFTWARE

Q: Your head of HR is excited. The programmer who manages your employee database has figured out how to update the employee directory automatically on your intranet by drawing information from a third-party payroll management program that your company has licensed. Not only will this development be useful for you, but also it will be useful for other companies' HR departments, because the program is very popular. "It took only a little bit of reverse-engineering to link the two together," the HR chief tells you. "That's ok, right?"

A: It really may be.

The first thing that you have to do is check the language of the third-party license. Most general software licenses expressly prohibit a licensee from reverse-engineering, disassembling, or decompiling the licensed software. If, however, you have lawfully obtained software and you reverse-engineer only the portion of the software that you need to create an interoperable program, not to duplicate or replace the licensed software, you may be able to perform limited reverse-engineering. In the 2000 case of *Sony v. Connectix*,¹⁶ the Ninth Circuit held that reverse-engineering for the purpose of creating both an interoperable software program and the intermediate copying necessary to get to underlying unprotectable elements of that software program constituted a fair use of the copyrighted materials.¹⁷ You also need to consider other laws, however, such as the Digital Millennium Copyright Act ("DMCA"),¹⁸ the Uniform Computer Information Transactions Act ("UCITA"),¹⁹ which Virginia and Maryland have adopted, and European Union ("EU") law.

The DMCA, the UCITA, and the EU

If you have agreed by a software license agreement or by another contract that you will not reverse-engineer the licensed software, this prohibition may be unenforceable as violative of public

policy. The DMCA and the UCITA seemingly lay the groundwork for such a finding. EU law is helpful in this area, as well.

DMCA

Under the DMCA, circumvention of copyright protection is allowable for certain reverse-engineering purposes. The DMCA extends copyright law protections to technological measures that control access to a particular portion of a software program. It explicitly provides, however, that "a person who has lawfully obtained the right to use a copy of a computer program" may circumvent such a technological measure.²⁰ The DMCA also provides the right to develop and employ technological means to circumvent a technological measure for the "purpose of enabling interoperability of an independently created computer program with other programs, if such means are necessary to achieve such interoperability"²¹ Legislative history suggests that reverse-engineering is not an exception under the DMCA when such information is readily available to the person engaging in the circumvention and to the extent that such action constitutes other copyright infringement.²²

UCITA

Among its provisions, the UCITA renders shrink-wrap and click-wrap licensing agreements enforceable, creates default choice of law/forum rules, establishes rules for electronic signatures, records, and authentication, expressly provides liability protection for software vendors, declares unenforceable contract terms that prohibit criticism, bans electronic self-help while other controls remain, and expressly authorizes reverse-engineering for interoperability purposes. Maryland and Virginia have adopted UCITA.²³ In those states, the statute codifies the public policy that contrary contract language—that is, prohibiting reverse-engineering—is unenforceable.

EU

The EU has also addressed reverse-engineering, alternatively called decompilation, and taken a somewhat liberal stance. A European Council directive holds that "[t]he authorization of the rightholder shall not be required where reproduction of the code and translation of its form . . . are indispens-

able to obtain the information necessary to achieve the interoperability of an independently created computer program with other programs”²⁴ Therefore, in the EU, a contract term prohibiting reverse-engineering for legitimate interoperability purposes would likely be unenforceable.

Practice Tip

Preventing competitors from developing competing and interoperable products is often a key part of software marketing strategy. To the extent that you are a licensor trying to prevent third parties from creating interoperable software components, consider avoiding an EU country or Virginia or Maryland as your choice of law. You may also wish to include a provision in your license requiring a licensee to inform you before it undertakes any reverse-engineering and to give you an opportunity to address its concerns another way.

PRESERVING A CLEAN ROOM

Q: The head of your client services group comes into your office with a triumphant smile. “We spent the past few weeks evaluating three software packages and came up with so many great ideas that, instead of choosing one of the vendors, we want to build our own program,” he tells you. “So you can just ignore those contracts I sent you to review. Skipping that review should save you a lot of time, right?”

A: That depends.

When companies need software to achieve specific functionality, they often must choose between licensing the software from third parties or building the software internally. As an accidental licensor, you will likely need to give noninfringement representations to your users and indemnify them from third-party claims. Therefore, it is important that you stay clean during program development, not only for your own protection, but also for the benefit of your future customers. In short, to stay clean, you want to keep the developers away from the evaluators.

Different personnel should have build or buy responsibilities, and to the extent that those individuals must interact, you should help to define the parameters of their communications. Whether you create a formal clean room environment or simply

try to avoid indicia of obvious taint, you should plan to minimize the likelihood of successful claims of copyright infringement, trade secret misappropriation, or unfair competition when your company is making a build or buy decision.

Clean room describes an environment in which developers work without access to third-party proprietary technology. To achieve a clean room, one personnel group develops high-level specifications that describe the functionality of the intended software product, but do not include the actual code or other copyrightable expression from the software that serves as the basis for the desired functionality. A second personnel group reviews the specifications and creates computer code that meets their requirements. The purpose of the clean room process is to help a company (1) develop technology that does not infringe another company’s copyright or misappropriate another company’s trade secrets and

CLEAN ROOM DESCRIBES AN ENVIRONMENT IN WHICH DEVELOPERS WORK WITHOUT ACCESS TO THIRD-PARTY PROPRIETARY TECHNOLOGY.

(2) maintain documentation that can serve as evidence that no such infringement or misappropriation has occurred. To analyze the status of your environment or to create a clean room environment from scratch, see the clean room checklist sidebar on page 63.

A clean room process can be useful because courts may find infringement liability even when a developer had minimal access to third-party technology. Courts may disregard the developer’s innocence under the concept of unconscious copying. If a company can demonstrate that the claimed proprietary trade secrets or copyrightable work in the third-party technology was not available to its developers, a third party would find it difficult, if not impossible, to prove a trade secret misappropriation or copyright infringement claim.²⁵

Nondisclosure or Confidentiality

You should also carefully review and consider any nondisclosure or confidentiality agreements

that your company might enter into with potential vendors. Businesspeople often take confidentiality agreements more lightly than other types of contracts, believing them to be a standard part of doing business, but such agreements create the appearance of misconduct. Their recitals suggest that every aspect (even functionality) of a third-party vendor's product is confidential and proprietary. Some confidentiality agreements even contain an obligation not to create the desired software or software that competes with the product that you are evaluating. To the extent that license agreements may attempt to place that restriction on your company, you should vigorously object. Indeed, courts have found a contractual obligation not to create competing software to be copyright misuse.²⁶

You may be able to include language in applicable nondisclosure agreements or evaluation licenses that expressly acknowledges that your company may be developing its own, similar product. The following is an example of such a clause:

Sample contract clause: Nothing herein shall prohibit or restrict the Receiving Party's right to develop, use, or market products or services similar to or competitive with those of the Disclosing Party, as long as such activity does not otherwise constitute a breach of this Agreement. The Disclosing Party acknowledges that the Receiving Party may already possess or have developed or market products, services, concepts, or ideas similar to or competitive with those of the Disclosing Party contained in the Confidential Information.

Practice Tip

If the software that you are evaluating is protected by patent(s), clean room development will not serve as a defense. Before undertaking a major new software development, you may wish to commission a prior art search for patents that your software may infringe, particularly if your company is in an industry in which patent protection is common.

HANDLING ENCRYPTION AND EXPORT ISSUES

Q: Your company is finishing work on a new inventory management program, and your IT manager wants to outsource completion of the security

CLEAN ROOM CHECKLIST

Use the following checklist to create a clean room, an environment in which developers work without access to third-party proprietary technology, as part of a plan to create noninfringing technology and supporting documentation:

- Identify managers and teams for both the evaluation specification writing process and the software development process. Keep the two groups separate.
- Prevent people at your company from executing software licenses, evaluation agreements, or confidentiality agreements with third parties without approval by your legal department.
- Isolate all third-party materials that you receive during the evaluation specification process in a single location and/or hard drive.
- Keep third-party protected material out of your specifications. Once the specifications have been appropriately screened and filtered, you may give them to the development manager to begin the development work.
- Create an archive of all third-party materials and then delete and destroy working copies after the evaluation and specification writing process is complete.
- Require evaluators to document each significant step of their work and to include the names of participating personnel and relevant dates.

and transaction processing components to a developer in India, primarily for cost reasons. The transaction processing code contains encryption technology to protect customers' credit card numbers and identities. The work will be based on code started last summer by a company intern from Libya who recently returned there. "We want to send all of the code to India over the internet tomorrow. Do you see any problem with that?" the manager asks.

A: Yes.

In addition to the basic security issue of sending valuable source code via a public network and the copyright ownership issues associated with using an

*From this point on . . .
Explore information related to this topic.*

ONLINE:

- ACCA's Ecommerce Committee, which has a listserv to join and other benefits. If you have questions about the Ecommerce Committee or other ACCA committees, contact the chair (listed in each issue of the *ACCA Docket*), or contact Staff Attorney and Committees Manager Jacqueline Windley at 202.293.4103, ext. 314, or windley@acca.com, or visit ACCA OnlineSM at www.acca.com/networks/ecommerce.php.
- Hans Albers, *Enforcing Shrink-wrap and Click-wrap Licence Agreements*, GLOBAL COUNSEL, 2002, VII(7), 27, available on *PLC Law Department* at www.practicallaw.com/a25509.
- Karen L. Boudreau, "An Introduction to Software Licensing," *ACCA Docket* 20, no. 9 (2002): 54-71, available on ACCA OnlineSM at www.acca.com/protected/pubs/docket/on02/software1.php.
- Free Software Foundation, at www.fsf.org.
- The GNU Project, at www.gnu.org.
- Jay Hollander, *What Happens When a Software Vendor Goes Bankrupt?* GigaLaw.com, at www.gigalaw.com/articles/2000-all/hollander-2000-05-all.html.
- Roberta L. Horton, "Trademark and the Internet: Some Basics and Recent Developments," available on ACCA OnlineSM at www.acca.com/protected/legres/intelprop/trademarkbasics.pdf.
- Open Source Initiative, at www.opensource.org.
- Duncan Sinclair and Tony Woodgate, *Transactions and Practices: Intellectual Property Transactions*, COMPETITION PRACTICE MANUAL, Global Counsel, 2002, available on *PLC Law Department* at www.practicallaw.com/a14481.
- Johanna I. Werbach, Kerry A. Krzynowek, and William F. Porter, "Top 10 Copyright Issues for the Digital Age," *ACCA Docket* 20, no. 1 (2002): 50-71, available on ACCA OnlineSM at www.acca.com/protected/pubs/docket/jf02/copyright1.php.

ON PAPER:

- Brandon Dalling, *Protecting against International Infringements in the Digital Age Using United States Copyright Law: A Critical Analysis of the Current State of the Law*, 2001 B.Y.U.L. REV. 1279 (2001).
- Shelly Rothschild, *How to Protect the Rights of a Licensee When the Licensor Files for Bankruptcy*, LEADER'S BANKRUPTCY STRATEGIST, vol. 17, no. 1 (Nov. 1999).

If you like the resources listed here, visit ACCA's Virtual LibrarySM on ACCA OnlineSM at www.aca.com/resources/vl.php. Our library is stocked with information provided by ACCA members and others. If you have questions or need assistance in accessing this information, please contact Legal Resources Manager Karen Palmer at 202.293.4103, ext. 342, or palmer@acca.com. If you have resources, including redacted documents, that you are willing to share, email electronic documents to Managing Attorney Jim Merklinger at merklinger@acca.com.

independent contractor in another country, developers must always be mindful of U.S. export controls surrounding software, especially software containing encryption technology and/or functionality. These controls will affect to whom software developers may sell their products and with whom they may work to further develop software.

To complicate matters, three government entities administer export controls:

- Department of the Treasury's Office of Foreign Assets Control ("OFAC"), which administers comprehensive trade and economic sanctions against certain countries, such as Cuba, Iran, Iraq, Libya, North Korea, and the Sudan, and against certain groups of individuals, including terrorists and narcotics traffickers.
- Department of State's Office of Defense Trade Controls, which administers export controls on defense articles, including technology, and defense services pursuant to the International Traffic in Arms Regulation ("ITAR").
- Department of Commerce's Bureau of Industry and Security ("BIS"), which administers export controls pursuant to the Export Administration Regulations ("EAR").²⁷

Determining which U.S. controls apply to your exports and reexports (to new users, new destinations, or new uses from those for which you received original export authorization) will depend on the following factors: (1) how the software is classified under the Commerce Control List ("CCL"), (2) the country to which the software is being exported (some countries are subject to export licensing requirements, even for low-level technology), and (3) the use to which the customer puts the software (some uses, such as military uses, give rise to licensing requirements). Before exporting or reexporting any software—which includes providing the software to foreign nationals located within the United States and posting the software on the internet for anyone to access—you must know whether you need an export license or some form of government notification or review and what, if any, reporting requirements apply.

Export Control Analysis

You should first ask whether your encryption software will be exported or reexported to Cuba,

Iran, Iraq, Libya, North Korea, the Sudan, or Syria, the so-called T-7 countries (the seven countries for which exports are controlled for antiterrorism reasons), or provided to foreign nationals of any of the T-7 countries, even if the individual is located within the United States. If you provided encryption software to your brilliant summer intern, you will need to obtain an export license from BIS or OFAC before the software's export or reexport, which includes not only actual shipments to the T-7 countries, but also access to the software by foreign nationals of the T-7 countries.

Although less likely in the accidental licensor context, you should also ask whether the encryption software constitutes a "defense article," that is, an item or technical data designated on the U.S. Munitions list²⁸ or an item or technical data designated or determined to be a defense article pursuant to § 120.3 of the ITAR, called "Policy on designating and determining defense articles and services." If the encryption software is a defense article, you generally need to obtain an export license from the Office of Defense Trade Controls, unless an ITAR exemption applies.

Your analysis for all other encryption software should usually center on the Export Administration Regulations. Fortunately, June 6, 2002, EAR revisions regarding encryption increased the number of encryption items that you may export or reexport without an export license. As a result, you generally do not need an export license, except for the T-7 countries, nationals of the T-7 countries, or persons designated in Part 744 of the EAR²⁹, but you do need to provide notification to BIS or have BIS review your software or register it for review.

Encryption software will qualify as mass market and will not require an export license before export or reexport, absent the noted exceptions. But it will require government notification, if it employs a key length less than or equal to 64 bits for the symmetric algorithm, or government review, if it employs a key length greater than 64 bits for the symmetric algorithm, if all of the following conditions exist:

- The software is generally available to the public by being sold, without restriction, from stock at retail selling points by means of over-the-counter, mail-order, electronic, or telephone-call transactions.

- The user cannot easily change the cryptographic functionality within the software.
- The software is designed for installation by the user without further substantial support by the supplier.
- Details of the items are accessible and will be provided, when necessary, upon request, to the appropriate authority in the exporter's country in order to ascertain compliance with the first three conditions above.³⁰

Other items that require only government notification before their export or reexport, with the noted exceptions, include encryption software using key lengths less than or equal to 56 bits for symmetric algorithms, 512 bits for asymmetric algorithms, and 112 bits for elliptic curve algorithms, beta test encryption software, and publicly available encryption source code, such as source code posted to the internet, and the corresponding object code resulting from the compiling of such source code.³¹

Your noncompliance with export controls can result in heightened scrutiny of your exports, seizure and/or forfeiture of your exports, your company's debarment, suspension, or ineligibility to obtain export licenses or other authorizations from a U.S. agency or the denial of all export privileges for up to 10 years, and civil fines and/or criminal penalties.³² Willful violations of the EAR can result in corporate criminal fines of up to the greater of \$1 million or five times the value of the exports and individual fines of up to \$250,000 or imprisonment for up to 10 years or both.

Practice Tip

By allowing the Libyan intern to participate in your project, your company likely violated U.S. encryption export regulations. You should always be aware of the country of origin of individuals in your IT department and make sure that none is a national of the T-7 countries. Before sending the project to developers in India—and depending on the strength of the encryption used and its purpose—you will likely have to notify the U.S. government, at minimum, and perhaps also get a full review of the proposed export. Your failure to comply could result in administrative, civil, and/or criminal penalties. If a member of your IT department raises free speech as a reason that it can export encrypted software, you should be aware that thus far no federal court has found any

export controls imposed upon source code to be unconstitutional.³³

PROTECTING YOUR INTELLECTUAL PROPERTY

Q: Your chief financial officer is ecstatic. “Hey,” she tells you, “my team has developed an internal auditing program that every business in America can use. We’re going to shop it around, starting with a booth at a big CPA conference in Philadelphia next week. Should we do anything before the show to protect it?”

A: Absolutely.

Many types of intellectual property protection are available for software, but most require tradeoffs between protection and confidentiality. As an initial step, you should confirm that your company owns the new developments. If your own employees did all of the development work, automatically making it work made for hire, you have no problem. If your company used independent contractors, you should obtain signed agreements from them, acknowledging their contributions as work made for hire or as fully assigned and assignable to your company. Seek retroactive assignments from all outside developers, if you must.

With ownership of the developments confirmed or resolved, you should turn next to the various intellectual property regimes that may apply to the development, including copyright, patent, and trade secret protection.

Copyright

Of the intellectual property protections, copyright has the lowest threshold. Copyright protection attaches the moment that the software is fixed in a tangible medium, assuming minimal creativity. Nonetheless, there are benefits to registering the work at the U.S. Copyright Office, including the ability to obtain statutory damages (even absent any actual damages) and attorneys’ fees in a copyright infringement action. Registration is also necessary to bring an enforcement action, although, in some courts, it is enough to have filed a copyright application for the infringed work.³⁴

To register your copyright, you must deposit some or all of the source code with the Copyright Office.

Special rules allow you to block or reduce deposit materials for code constituting trade secrets. Moreover, although competitors may gain access to deposit materials, they may obtain copies only with your consent, in connection with litigation or subject to a court order. Bear in mind that, although copyright will protect literal or nonliteral copying of the software, it will not protect the basic functionality of the software. The scope of copyright protection for nonliteral copying is uneven and hard to predict.

Patent

You should also consider whether the software might be patentable. In the United States, if you do not apply for a patent within one year of use or publication, you will lose your ability to obtain patent protection for the software. Seemingly innocuous acts, such as filling in a date of first publication on a copyright application, can have a critical effect on patentability, because such acts might constitute evidence of commencement of the 12-month statutory bar. In most countries outside of the United States, you must file patent applications before any use or publication. Therefore, you must make the decision early about whether to file for a patent.

As for confidentiality, you should be aware that the U.S. Patent and Trademark Office (“PTO”) now publishes patent applications, which include a full disclosure of the invention, 18 months after filing. Applicants may request that their applications not be published, but they then forego the right to file for foreign patent protection. The public may access and obtain copies of issued patents. Therefore, once the patent has been published, trade secret law does not protect it. At this point, you must hope that a valid patent issues and is reasonable in scope of protection. Patent protection can be a powerful sword—or shield because patent cross-licensing is frequently used to resolve infringement claims.

Trade Secret

Of course, not all software will meet the stringent prerequisites for patent protection, and trade secret protection may be a viable alternative. Almost any subject matter that is valuable by virtue of not being generally known can qualify as a trade secret. Protection lasts only as long as the owner exercises reasonable efforts to maintain secrecy. You need to be able to show that you treated the

software as a trade secret before it was misappropriated. Trade secret protection is often the least bureaucratic and most inexpensive method of intellectual property protection.

Remember, however, that a trade secret is valuable only to the extent that you protect it. Regular use of confidentiality agreements with those who will have access to your trade secret, restrictive labeling, secure areas, passwords, tracking of copies, technical protections, and other indicia of secrecy may pay off.

Practice Tip

Make sure that your product development teams take appropriate precautions to keep their inventions private until your company can make an informed decision about whether to apply for copyright or patent protection.

DEALING WITH A VENDOR’S BANKRUPTCY

Q: Your CFO now tells you that the accounting software package that she plans to market incorporates some software from a third-party vendor who will soon be filing for bankruptcy and has said that it does not intend to carry out its maintenance obligations. “This situation might make it difficult for us to maintain our own product,” she tells you. “Do we have any rights or remedies?”

A: Yes.

Source Code Escrow

Source code escrow, which requires storing the source code with a trusted third party and usually in a secure, safety deposit type vault, is particularly important when you license software from a vendor on a perpetual basis for a one-time paid-up fee. Without such an escrow arrangement, the large license fee that your company paid upfront may suddenly become a writeoff, if the vendor goes bankrupt or decides not to maintain the software any longer.

Not all escrow agreements are alike, however. Be sure that the vendor’s failure to provide support and maintenance for the software in a timely fashion is a release condition. If the vendor’s financial distress is the only release condition, however, your

company may not have recourse for an ordinary, but potentially very harmful, breach of its maintenance obligations.

In addition, you should include in the escrow vault any programmers' notes—including the names of key employees who worked on the software, along with their contact information—and any other documentation that a third party unfamiliar with the software would need to use the source code. If the vendor has customized software for you, be sure that the customized version and not the standard, unmodified version is in the vault. Further, ensure that your escrow agreement expressly allows third parties to assist you in working with the source code because your IT department may not have the expertise or time necessary to perform maintenance tasks or to fix errors in the software itself.

Finally, as discussed below, the escrow agreement should specifically refer to § 365(n) of the U.S. Bankruptcy Code in order to take advantage of certain protections in the event that the source is released because the licensor has declared bankruptcy.³⁵ The escrow agreement should state that it is a "supplement" to the license agreement, as that term is used in § 365(n).³⁶ In addition, ideally, the escrow agreement should be structured as two separate agreements, one between the licensor and the escrow agent and one between the escrow agent and you, the licensee, to avoid the risk that the licensor/vendor might try to "reject" the escrow agreement in addition to the license agreement.

Section 365(n)

Suppose that your company hasn't obtained source code escrow for the accounting software or, as often happens, that your IT department doesn't have the resources needed to maintain the abandoned software itself on an ongoing basis. You must turn to § 365(n) of the bankruptcy code for answers.

When a debtor licensor files for bankruptcy, it has the right to decide whether it will continue to meet obligations under executory contracts entered into before the bankruptcy or, instead, to reject them. Software licenses typically would be executory, because both parties would have continuing material obligations. If an intellectual property licensor rejects an executory contract with a licensee, the licensee has two choices under § 365(n).

Under the first option, if the licensor's rejection would cause a breach of its obligations outside the bankruptcy context, the licensee could declare the contract terminated and make a claim for damages. Licensees generally do not favor this option because they do not secure the continued right to use the licensed software going forward and thus, as unsecured creditors, they are unlikely to be able to collect significant damages.

The second option allows the licensee to demand from the licensor the continued ability to use the intellectual property through the termination or expiration of the agreement and any extensions that the licensee has as of right. This request must be made in writing. The licensee must still pay all ongoing royalty payments, which the code broadly defines, required under the contract to continue to use the intellectual property. If payments under the agreement combine license and maintenance fees, the licensee must continue to pay this lump sum, even if the licensor has rejected the contract and is, therefore, not required to perform any additional maintenance obligations.³⁷ To sidestep this pitfall, you should (1) avoid using the word "royalties" in describing payments to a vendor/licensor and (2) carefully itemize all fees paid to a licensor—that is, distinguish maintenance fees from license fees from service fees and so forth—to minimize the amount of fees attributable to the software license.

You may also want to consider inserting provisions into the license agreement to dissuade licensors from rejecting the license agreement in the first place. A liquidated damages clause that provides for significant damages in favor of the licensee, for example, may cause the licensor to think twice about rejecting it.

Practice Tip

If you have serious concerns about a vendor's financial condition at the time of signing a software license and the license is exclusive, you should consider taking and perfecting a security interest in the software. Further, you should be aware that § 365(n) of the U.S. Bankruptcy Code expressly excludes trademarks. Thus, if you market software under a licensor's trademark and the licensor declares bankruptcy, you may have to stop using the trademark in the event that the licensor rejects the contract. For this reason, if a software license is exclusive and your continued use of the licensor's brand name is as important

as the software itself, consider taking a security interest in the trademark, as well as the software.

CONCLUSION

The foregoing software licensing and development-related issues may not always be critical to a licensee, but they may become material if your company becomes an accidental licensor. As a licensor, you will want to know that you can modify, distribute, and protect your code. Government contract and export issues that you may gloss over when you are just the user of software may be more relevant when you are developing your own product.

Because you never know when your company may become a licensor, you should pay attention to the development process. Be sensitive to clean room scenarios. Your communication with internal clients about proper system development may avoid the creation of good products with bad facts. In some situations, however, your programmers' instincts may be correct. Notwithstanding an express prohibition on reverse-engineering, you may decide to live dangerously and permit your IT team to do some decompiling. Finally, consider the financial strength of your vendor-licensors and be as ready as possible in the event that they fail to support their products or attempt to reject your licenses.

Becoming an accidental licensor can be a lucrative opportunity for your company, as long as legal risks and pitfalls associated with licensing software are identified and minimized. To do this, in-house counsel need to think carefully through the issues surrounding development and licensing of the software the same way that a licensing powerhouse like Microsoft® or Oracle® would. Whatever your company's core business may be, adopting and using best practices in developing and licensing software should be no accident. ■

NOTES

1. OSI is a nonprofit group that advocates licensor source code in the manner described above.
2. See www.opensource.org/docs/def_print.php.
3. See www.opensource.org/licenses/index.php.
4. FSF is the alternative open access software advocacy group. The FSF's website at www.gnu.org discusses the GNU Project.
5. The FSF's four basic freedoms are (1) the freedom to run the program, for any purposes, (2) the freedom to study how the program works and to adapt it to your needs with access to source code a definite precondition, (3) the freedom to redistribute copies so that you can help your neighbor, and (4) the freedom to improve the program and to release your improvements to the public so that the whole community benefits. See www.gnu.org/philosophy/free-sw.html. You should receive all of the foregoing rights without subsequently having to ask or pay for permission.
6. The FSF website provides further guidance about what constitutes aggregation and what constitutes combining two modules into one program. Ultimately, however, a court may have to decide. See www.fsf.org/licenses/gpl-faq.html.
7. *Id.*
8. See DFARS 227.7203-5 and 252.227-7014.
9. DFARS 227 7203-10 (c)(1).
10. DFARS 252.227-7025.
11. DFARS 252.227.7014 (a)(10)
12. See DFARS 227-7203-10(b)(1).
13. You can find the GPR legend at DFARS 252.227-7014(f)(2) and the restricted rights legend at DFARS 252.227-7014(f)(3).
14. See DFARS 252.227-7014(f)(1).
15. See *id.*
16. 203 F.3d 596 (9th Cir. 2000).
17. *Id.* at 607-08. See also 1 ROGER M. MILGRIM, MILGRIM ON TRADE SECRETS § 1.01[2][a], at 1-34 (1996).
18. 17 U.S.C. § 1201(f) *et seq.* (2002).
19. See Va. Code Ann. § 59.1-501.5 *et seq.* (Michie 2002); Md. Commercial Law Code Ann., [Com. Law II] § 22-105 *et seq.* (2002).
20. 17 U.S.C. § 1201(f)(1).
21. 17 U.S.C. § 1201(f)(2).
22. H.R. Rep. No. 105-551 pt. 2, at 42 (1998). See also Brandon Dalling, *Protecting against International Infringements in the Digital Age Using United States Copyright Law: A Critical Analysis of the Current State of the Law*, 2001 B.Y.U.L. REV. 1279 (2001).
23. Va. Code Ann. §59.1-501.5, citing Restatement (Second) of Contracts § 178 (1981); Md. Commercial Law Code Ann., § 22-105(b).
24. Council Directive 91/250/EEC of 14 May 1991 on the legal protection of computer programs, at www.europa.eu.int.
25. See generally *Computer Assoc. Int'l v. Altai, Inc.* 982 F.2d 693 (2d Cir. 1992, *aff'd* 775 F. Supp. 544 (E.D.N.Y. 1991)); *NEC Corp. v. Intel Corp.* 10 U.S.P.Q.2d 1177 (N.D. Cal. 1989).
26. See, e.g., *Lasercomb America, Inc. v. Reynolds*, 911 F.2d 970 (4th Cir. 1990).

-
27. Relevant export information can be found on the internet at the following websites: www.treas.gov/offices/enforcement/ofac, www.pmdtc.org, www.bxa.doc.gov.
 28. See § 121.1 of the ITAR.
 29. Part 744 of the EAR describes the export controls that apply to certain end users and uses. See EAR §§744.9, 744.13, 744.14, and 744.15.
 30. See EAR § 742.15 (b)(2).
 31. See CSEE EAR §§ 740.9 (c)(8), 740.13(e), 740.17(d)(3), and 742.15(b)(1).
 32. See CFR chapter 5, Part 127 of the ITAR, and Part 764 of the EAR.
 33. See *Bernstein v. United States Department of Justice*, 176 F.3d 1132 (9th Cir.), *reh'g granted and opinion withdrawn*, 192 F.3d 1308 (9th Cir. 1999); *Junger v. Daley*, 209 F.3d 481 (6th Cir. 2000). *Cf.* *Universal City Studios, Inc. v. Corley*, 273 F.3d 429 (2d Cir. 2001); *Universal City Studios, Inc. v. Reimerdes*, 111 F. Supp. 2d 294 (S.D.N.Y. 2000).
 34. Section 411 (a) of the Copyright Act establishes registration as a prerequisite for an infringement action, but the Fifth Circuit considers this requirement satisfied by merely filing an application for registration. The Second Circuit and the Eleventh Circuit require an actual certificate of registration (although one S.D.N.Y. Court, citing *NIMMER ON COPYRIGHT*, accepted a pending application). There is a split within the Ninth Circuit and the D.C. Circuit as to whether a certificate or an application is required to bring suit. Most recently, in *Strategy Source v. Lee*, 2002 WL 316 49982 (DDC), the District Court of DC reaffirmed that a registration was necessary. (This case has many relevant cites on this topic).
 35. USC § 8101(52)-(53), 365 (i)(1994) (the Intellectual Property Act.) The Intellectual Property Act amended the Bankruptcy Code to add a new subsection 8365(n) which provides licensees of intellectual property limited rights to continued use of the intellectual property and certain other rights in the event that the debtor/licensor declares bankruptcy.
 36. See Shelly Rothschild, *How to Protect the Rights of a Licensee When the Licensor Files for Bankruptcy*, LEADER'S BANKRUPTCY STRATEGIST, vol. 17, no. 1 (Nov. 1999).
 37. See Jay Hollander, *What Happens When a Software Vendor Goes Bankrupt?* on GigaLaw.com, at www.gigalaw.com/articles/2000-all/hollander-2000-05-all.html.
-