



A Practical Guide to License Conflicts

Jeffery S. Norman, Partner

Kirkland & Ellis LLP

March 2008

Topic Outline

- Introduction
- Identification
- Analysis
- Mitigation

INTRODUCTION

License Conflict Defined

License Conflict Is:

- Loss of rights under license triggered by another agreement
- Inability of licensee to comply with both terms of license and terms of another agreement
- Breach of license triggered in part by another agreement
- Imposition of additional obligations or restrictions due to another agreement

License Conflict Is Not:

- Merely license compliance
- Merely license compatibility
- Conflict between preserving a licensee's own trade secrets and compliance with GPL/copyleft conditions
- Failure to properly provide required notices or attribution
- Limited to open source software

The Basics: DO

- Manage license conflict review as a legal process
- Identify and collect existing information from key touchpoints, historians, information maintainers, records
- Review existing OSS practices
- Identify and understand products and systems in which products are used
- Read every license
- Ensure that internal communications are made to legal staff, and clearly marked as privileged attorney-client communications

The Basics: DON'T

- Rely on scanning alone
- Wait until product release to review OSS usage and licensing
- Relegate license conflict review to engineering
- Relegate license conflict review to legal
- Treat OSS issues as religious issues
- Limit license conflict review to company products
- Limit license conflict review to OSS
- Create documents that might appear to admit breach or infringement

Myths

1. If you don't distribute software, there can't be a license conflict
2. If you only dynamically link and limit use to LGPL and BSD style licenses, there can't be a conflict
3. Banning GPL is sufficient
4. There are no real legal risks from use of OSS

Themes

- Think outside the product box
 - Taking a systems approach
 - Review all the software touchpoints
- Understand context
 - Clearly define the purpose for the license conflict review
- Practical beats perfect
- Don't panic

IDENTIFICATION

Collect and Identify

- Cast a wide net
 - Identify touchpoints
 - Identify information repositories
 - Obtain records
 - Collect and review licenses
- Determine origins of code (pedigree)
 - Especially difficult with licensed or acquired code
 - Obtain historical agreements as they may create potential license conflicts
- Trust but verify – the role of scanning

Inbound Touchpoints

- Acquisition
 - In-licensed
 - M&A/purchased IP
- Joint ventures and collaborations

Outbound Touchpoints

- Distribution
 - Final product/SKUs
 - Alpha, beta releases
 - Patches
 - Developer assistance (SDK, toolchains, emulators, APIs, sample code, templates ...)
 - M&A/IP asset sales
 - Joint ventures and collaborations
 - Third party manufacturers

Development Touchpoints

- Development stages
 - Design
 - Build
 - Test
 - Release
- Loci of Development
 - R&D centers
 - Outsourced development
 - Community development (open and closed)

ANALYSIS

Sources of Conflict

- Major sources of license conflict:
 - Disclosure mandates
 - Application of license terms or mandates to derivative works
 - Anti-conflict terms
 - Ambiguity
 - Retroactive terms

Basic Steps

- Analyze core licenses with:
 - *Capacity for Conflict*, and
 - *Red Zone Conflict Terms*
- Apply external risk filters
- Conduct thorough analysis
- If appropriate expand scope of review to Orange, Yellow

Capacity for License Conflict

- A license with capacity for conflict may conflict with any license, not just other licenses with capacity
- Licenses without capacity for conflict cannot conflict with each other
- Classify conflicts based on potential for severity before reviewing individual conflicts

Capacity for License Conflict

- A license has **capacity for license conflict** if it contains:
 - Disclosure Mandates: Requirement that source code for modifications or derivative works must be disclosed
 - License Recursion: Restricts the choice of license for distribution of any new or derivative work
 - Retroactivity: Permits retroactive application of new terms
 - Trumping: Purports to limit or restrict application of the terms of another agreement

Capacity for License Conflict

- (... continued):
 - Penalties: May impose any condition, obligation or restriction, or that may limit or negate any right, when licensed code is used/distributed together with code licensed under other terms
 - Conditional License: Terms that condition the applicability of the copyright license grant upon compliance with other terms

License Conflict Zones

- Basic idea
 - Identify types of conflict most likely to have material impact in general
 - Classify potential severity (red, orange, yellow)
- Heuristic approach
 - Each situation is different
 - Classification depends on context

Red License Conflicts

- Frequently presents severe conflict issues
 - Disclosure Mandates
 - License Recursion (when applicable to derivative works in general)
 - Conditional License (where condition relates to terms of other license)
 - Retroactivity

Orange License Conflicts

- Sometimes presents moderate to severe issues
 - License Recursion (when limited to modified files and not applicable to derivative works in general)
 - Penalties that relate to IP rights

Yellow License Conflicts

- Issues are not typically severe and may be more easily corrected
 - Trumping
 - Penalties limited to loss of non-specific patent non-assertion protection
 - License Recursion (when limited to original work only)
 - License Recursion (when limited to contribution to “official” source only)

Analysis of Specific Conflicts

- Once actual license conflict is identified, contextual risk analysis should be performed
 - Focus on outcome determinative factors
 - Where legal judgment or factual circumstance is unclear, assume the negative outcome
 - Make legal judgments to eliminate chafe
 - Dispense with weak theoretical outlier risks
 - Ensure that legal judgment is consistent
 - But don't confuse lack of historical precedent with strength of case
 - Analyze risk in context of contemplated action
 - Prospective analysis should focus on incremental risks only

Apply Filters

- Segregate “fundamental risks”
 - Risks that cannot be realistically mitigated without fundamental alteration of material part of historical business
 - Fundamental risks must be considered in M&A/IP asset acquisitions and long term strategy
 - Fundamental risks should be disregarded in most other contexts

MITIGATION

Strategies: Existing Risks

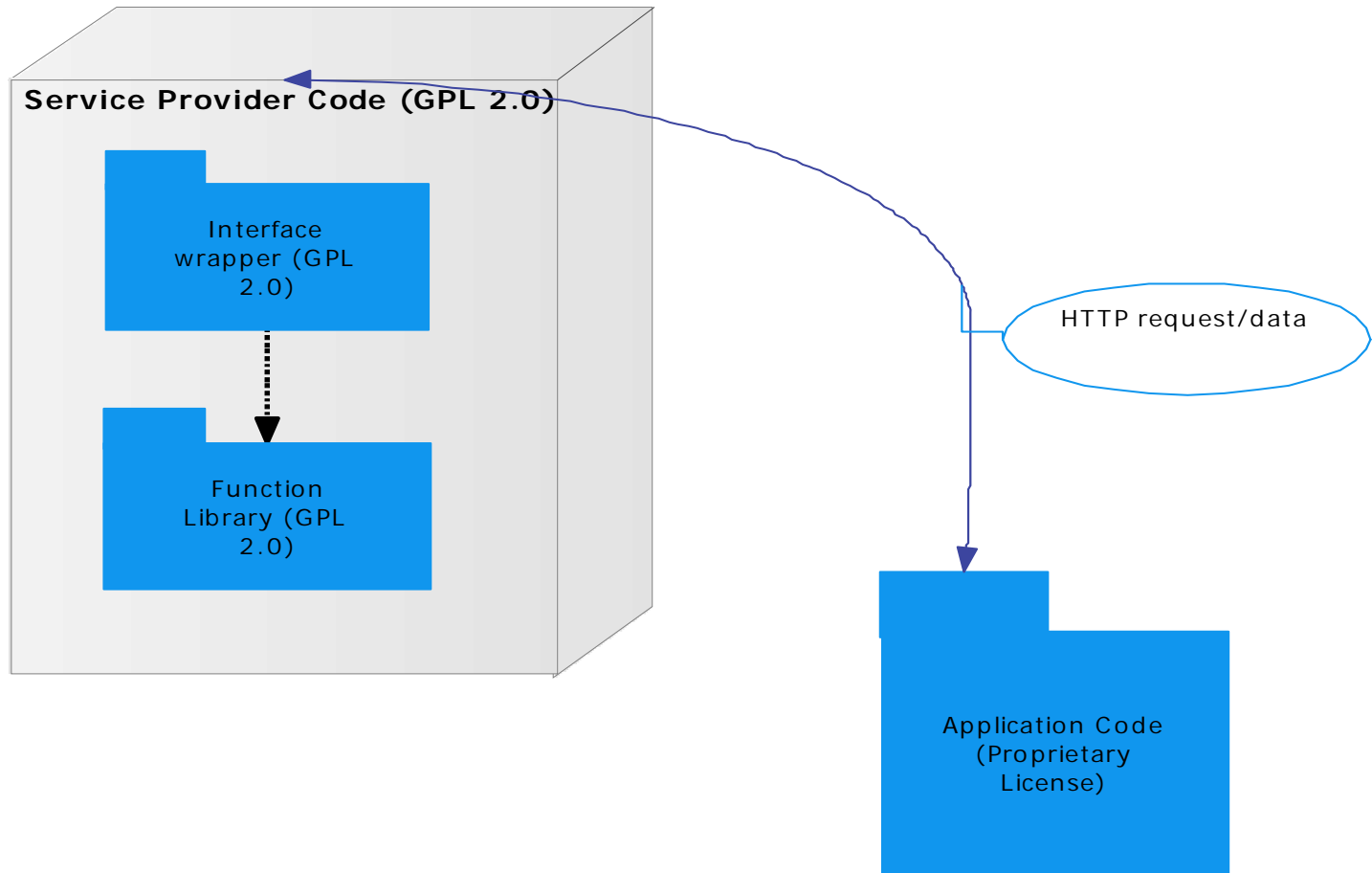
- Consider potential risks from mitigation strategies
 - Especially where most product already in channel; incremental risk is small relative to pre-existing risk
 - Some mitigation strategies may end up providing notice to a potential plaintiff
 - Mitigation strategy may be used as evidence of non-compliance

Strategies: Existing Risks

- Dual licensing
- Replace
 - OSS component
 - Proprietary component
- Rewrite
 - Subject to clean room requirements
- Obtain sufficient rights to avoid conflict

Strategies: Existing Risks

- Re-architect to avoid distribution of derivative work
 - Many copyleft licenses depend on distribution of derivative work; strategy avoids distribution of derivative work
 - Distributed product cannot include any derivative work of code subject to the applicable conflicting license (watch use of headers and APIs)
 - Recipient must obtain underlying OSS code
 - Ensure that recipient has license right sufficient to create derivative work without conflict or breach (or else contributory infringement and interference with contract may apply)
 - Proprietary and OSS code can be used side by side if separated and made interoperable
 - Firewall using http should be uncontroversial
 - Firewall using intermediate API code:
 - Firewall program itself is subject to copyleft (e.g., GPL) but provides interface used by programs whose licenses would otherwise conflict; interface must not be derivative work based upon underlying copyleft work
 - Use of API firewalls can generate negative reaction – e.g., recent kernel discussions concerning ndiswrapper



Firewall Technique to Avoid Potential License Conflict

Strategies: Existing Risks

- Obtain indemnification from responsible party
 - May be feasible where responsible party's code is source of potential conflict
 - Responsible party must be able to satisfy the potential claims
- Phase-out product
- Manage public response
 - Don't leave inquiries to outsourced tech support!
- Consider "lesser" breach
 - Where terms of license can be satisfied by breaching another agreement, within control of licensee
 - Consequences of breach of conflicting license must be quantified
 - Benefit must outweigh such consequences

Strategies: Prospective

- Address potential conflicts before development
 - Conduct diligence on inbound software (licensed, M&A) before substantial reliance
 - Establish legal and technical OSS review team
 - As resource
 - As required approval at designated touchpoints
 - Conduct periodic scans to verify

Strategies: Prospective

- Use cases
 - Helpful to establish policy and consistency
 - Education
 - Guidance for code review
 - Documentation and privilege issues
 - Defeasibility