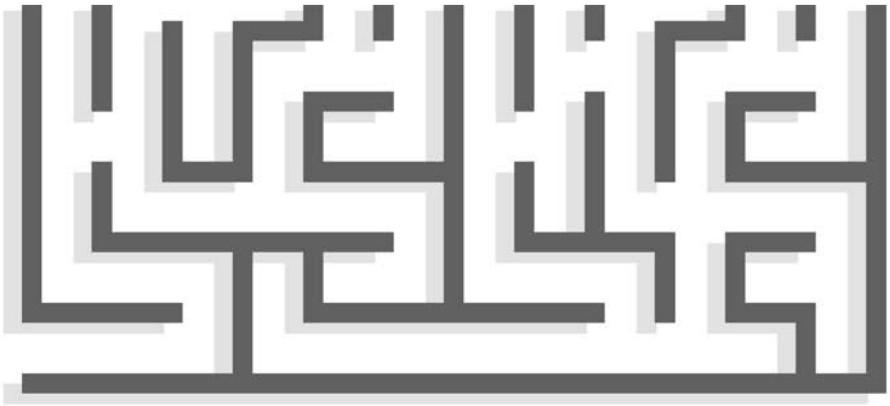
企業における
オープン ソース
コンプライアンス



THE LINUX FOUNDATION

Ibrahim Haddad, PhD

このページは意図的に空白にしてあります。

Ibrahim Haddad, Ph.D.

企業における
オープン ソース コンプライアンス

The Linux Foundation

2016

目次

第 1 章 オープン ソース コンプライアンス入門 16

変わりゆくビジネス環境 16

オープン ソースコン プライアンス手始め 19

オープン ソースのコンプライアンスを保証する利点 20

コンプライアンスの失敗 21

知的財産権上の失敗 22

ライセンス コンプライアンス問題 24

コンプライアンス プロセス上の失敗 26

得た教訓 28

製品出荷やサービス立ち上げに先立ち コンプライアンスを確実に行う 28

コンプライアンス違反は高くつく 29

関係は大事 30

訓練は重要 30

第 2 章 オープン ソース管理プログラムの確立 31

オープン ソース コンプライアンス プログラム 31

コンプライアンス戦略 32

照会応答戦略 32

ポリシーとプロセス 32

コンプライアンス チーム 33

ツール 34

ウェブ上のプレゼンス 35

教育 36

自動化 37
メッセージの発信 37

コンプライアンスの難しさと解決策 37

長期的な目標と短期的な実行 39
コンプライアンスについての対話 40
クリーンなソフトウェア起点 の確立 41
コンプライアンスの維持 42
内面化と末永い実行 43

第 3 章 コンプライアンス達成に向けて： 役割と責任 46

オープン ソース評価委員会 (Open Source Review Board : OSRB) 50
法務 53
エンジニアリングおよび製品チーム 55
コンプライアンス オフィサー 57
オープン ソース幹部会議 (OSEC) 58
文書化 58
ローカル化 59
サプライチェーン 59
IT 60
企業買収 60

第 4 章 オープン ソース コンプライアンス プロセス 62

効果的なコンプライアンス 63
一貫したコンプライアンス プロセスの要素 64

ステップ 1 — オープン ソースの特定 65
ステップ 2 — ソース コードの監査 67
ステップ 3 — 課題解決 70

ステップ 4 — レビュー 70
ステップ 5 — 承認 72
ステップ 6 — 登録 73
ステップ 7 — 通知 74
ステップ 8 — 頒布前検証 75
ステップ 9 — 頒布 76
ステップ 10 — 最終検証 76

第 5 章 コンプライアンス プロセスとポリシー 78

ポリシー 78
プロセス 79

ソース コード スキャン 79
特定と解決 81
法務レビュー 81
アーキテクチャー レビュー 82
最終レビュー 83

プロセスの各ステージにおけるインプットとアウトプット 83

ソース コード スキャン段階 84
特定と解決フェーズ 85
法務レビュー 85
アーキテクチャー レビュー 87
最終承認フェーズ 87

詳細な利用プロセス 88
インクリメンタル コンプライアンス プロセス 93
OSRB 利用フォーム 95

OSRB 利用フォームに影響する規則 99

監査 99

ソース コード頒布 100

頒布の動機 100
頒布プロセスとポリシー 101
頒布方法と様式 103
頒布チェックリスト 104
頒布前チェックリスト 105
一般公開後のチェックリスト 107
書面による申出 107

第 6 章 コンプライアンス プロセス管理の 推奨プラクティス 109

コンプライアンス プロセス 109

特定フェーズ 109
ソース コード監査 111
課題解決 112
レビュー 113
承認 114
通知 115
検証 115

ツールと自動化 116

ソース コード特定ツール 117
プロジェクト管理ツール 118
ソフトウェア BOM 差分ツール 118
リンク解析ツール 119

第 7 章 コンプライアンスに関する照会の管理 121

コンプライアンスに関する照会への対応 122

確認 122
通知 123

調査 123
報告 123
照会のクローズ 124
矯正 124
改善 124
一般的な考察 124

第 8 章 その他のコンプライアンス関連プラクティス 125

従業員の評価 125
ウェブ ポータル 126
意思伝達 126
トレーニング 127
非公式トレーニング 127
公式トレーニング 128
ソース コード変更に関する考察 128
通知に関する考察 128
頒布に関する考察 129
利用に関する考察 130
帰属についての考察 132
帰属タイプ 132
帰属の提示 133
特定のライセンス義務 133
一般的なガイドライン 135

第 9 章 オープン ソース法務サポートを拡大させる 137

実際的な法務アドバイス 137
ライセンス プレーブック 138

ライセンス両立性マトリックス 139
ライセンス分類 141
ソフトウェア相互作用法 143
チェックリスト 145
結論 146

序文（Preface）

オープン ソース コンプライアンスに携わるようになったのは、ソフトウェア開発者として私のキャリアが始まったころでした。そこから今に至る 20 年、オープン ソース コンプライアンスは直接的にも間接的にも私の仕事の一部であり続けています。オープン ソース ソフトウェアと共に歩んできた私の旅ですが、振り返るとオープン ソース コンプライアンスにおける実践的（プラクティカル）な情報を見つけるのに苦労した旅でした。そういったこともあり私自身の経験を公開することに興味が湧いてきました。私の経験を公開することでいろいろな人たちが何かを学び、そして学んだ人が経験したことを公開するようになり、さらに私たち皆が産業において製品デリバリの期間やエンジニアリング リソースに与えるインパクトを最小限にとどめるよう、オープン ソース コンプライアンスを実現する、よりよいやり方に向けた取り組みに尽力できるのではないか、と。

このハンドブックは、まさにそういった企業でオープン ソース コンプライアンスを推進してきた私の経験をまとめたものであり、オープン ソース コンプライアンス プログラムの整備、その運営についての実践的側面にフォーカスしたものとなっています。ただ私の経験の大半が（C や C++が主要プログラミング言語である）組み込み領域中心だったため、本ハンドブックでは全般的にその傾向が色濃く出ている点についてはご留意いただければと思います。

皆さんがオープン ソース コンプライアンスを実現するべく、日々の推進活動で本書が役立ってくれることを願ってやみません。

序文（Foreword）

オープン ソースは、ソフトウェアや知的財産に関わる個人によってリードされた理想主義的な運動からばかりでなく、オープン ソースが IT 戦略における主要な部分であると認識して、オープン ソース開発に参加したいと考えているような（政府、企業、大学などの）組織における活動によっても、広がってきています。Linux や他のオープン ソース技術における初期段階の成功は、技術の全ての分野にも広がってきています。

伝統的な組織も注目しています。オープン ソース ソフトウェアを優先度の高い項目に挙げ、ソフトウェアを組織活動で戦略的優位を得るために使っています。

エンタープライズ IT でのオープン ソースの利用は、2010 年から倍増しています。

調査した企業の 78%は、オープン ソース上でビジネスを運用しています。

64%が、現在、オープン ソース プロジェクトに参加しています。

39%が、自社のオープン ソース プロジェクトを立ち上げています。

ノースブリッジ&ブラックダック
「オープン ソース未来予測 2015」

「オープン ソース ファースト：納税者のお金で開発された全てのソリューションは、納税者のものに（オープン ソース）」GSA において開発された全てのコードは、オープン ライセンスのもとに共有されるべきであり、他の人々はそれから便益を受けることができます。加えて、自分たちが設計したソリューションとしてのオープン ソース ソフトウェアを優先的に利用していきます。

CIO オフィス 米国連邦調達庁
（年間 660 億ドルの調達を行う米国政府機関）

「ブロックチェーン技術の開発は、金融サービス業界の業務や経済構造を再定義する可能性を秘めています。業界が、もっと積極的に、もっと効率的に、もっとディジタル化を促進しようと努力している、そういう重要な時期にオープン ソースは現れています。オープン ソース開発は、イノベーションを加速させ、この技術の大規模な利用の促進を助けてくれるでしょう。我々は Hyperledger プロジェクトをサポートしていることに誇りを感じています。」

リチャード ラム（Accenture 社 財務サービス担当チーフ エグゼクティブ）

「成長し活気ある開発コミュニティへのメンバー投資から、Dronecode プロジェクトの初年度はかなりエキサイティングなものです。共通プラットフォームを確立するために共に努力し、オープン ソース ベスト プラクティスを利用することで、カメラからクラウドにまで広がるドローン アプリケーションの基盤を確立することができます。Dronecode の「フルスタック」プラットフォーム アプローチは、メンバーのハードウェアとソフトウェアのイノベーションを結合しており、自律的で、環境を認識し、継続的にネット接続されている新世代ドローンを作り出すことでしょう。それは、飛行する IoT です。」

クリス アンダーセン（3DR 社 CEO）

（ワイアード マガジン 前編集長、「ロング テール」著者）

「オープン ソースは、我々の開発に本質的なものです。偉大なソリューションを構築するために、人々を共に作業させるパワフルなアプローチです。そこからは実際にメリットが共有されます。」

ロブ アレキサンダー（Capital One 社 CIO）

組織は、オープン ソース コミュニティへの適切な参加や、法的で責任あるやり方での参加の最善な方法についてのガイダンスを探しています。参加者は、コードと IP を共有することを望み、IP 資産（商標、著作権、特許）のための、信頼のおける中立的な管理場所を必要としています。リソース プール（財政、技術）のフレームワークを必要としています。

参加者は、効果的な方法で、自分の競争者といかに協力するかについてのトレーニングを必要としています。そのためにも、本書は、オープン ソース ライセンスの精神と法的項目を固守しながらも、共有される価値とイノベーションを創造する最善の方法について共有理解を作り出すことに焦点を合わせています。

このページは意図的に空白にしてあります。

第 1 章

オープン ソース コンプライアンス入門

変わりゆくビジネス環境

従来、プラットフォームやソフトウェア スタックはプロプライエタリなソフトウェアを使って実装され、内部開発されたソフトや交渉の結果であるライセンス条件によるサードパーティのソフトから成るさまざまなソフトウェアのブロックから構成されていました。ビジネス環境は予測可能で、企業は潜在的なリスクをソフトウェア ベンダーとのライセンス交渉や契約交渉を通じて軽減していました。全てのソフトウェア コンポーネントについて誰が提供者であるかを知るのは大変容易でした。図 1 は従来のハードウェア、ソフトウェアのプラットフォームについて主なブロックを示したものです。

図 1：プロプライエタリなソフトウェアのブロックに依る従来のソフトウェア プラットフォームの単純化したアーキテクチャー

時と共に企業はオープン ソース ソフトウェアを自社のプラットフォームやソフトウェア スタックに組み込み、その恩恵に与かるようになってきました。その理由は製品ごとにさまざまですが、さまざまな業界で共通するのは、オープン ソースのコンポーネントには即座に使える卓越した特徴があったこと、分散的な開発による市場投入への時間短縮により経済的に有意の利益があったこと、そしてソース コードをカスタマイズするという新しく出来ることを提供したこと、です。その結果、複数のソースによる新たな開発モデルが登場しました。その結果、複数のソースによる新たな開発モデルが登場しました。

この新たなモデルでは、製品は下記の任意の組み合わせとなります。

- その製品やサービスを作る企業が開発したプロプライエタリ コード
- 元々はその企業によりオープン ソース ライセンス下でオープン ソース コンポーネントを統合したり適用したりすることで開発されたが、上流のオープン ソースプロジェクトに寄付されず戻されなかったプロプライエタリ コード
- サードパーティ ソフトウェア プロバイダーにより開発され、製品やサービスを作る企業が商用ライセンスの下で受領したサードパーティの商用コード
- オープン ソース コミュニティにより開発され、製品やサービスを作る企業がオープン ソース ライセンスの下で受領したオープン ソース コード

図2（次ページ）に複数のソースによる開発モデル、および入ってくるソース コードのさまざまな組み合わせを示します。

この開発モデルでは、ソフトウェア コンポーネントは任意の数の出所から来た、さまざまなライセンス下でライセンスされたソース コードから構成されえます。たとえば、ソフトウェア コンポーネント A はサードパーティのプロプライエタリ ソース コードに加えプロプライエタリ ソース コードも含んでおり、ソフトウェア コンポーネント B はオープン ソースプロジェクトからのソース コードに加えプロプライエタリ ソース コードを含む、などです。

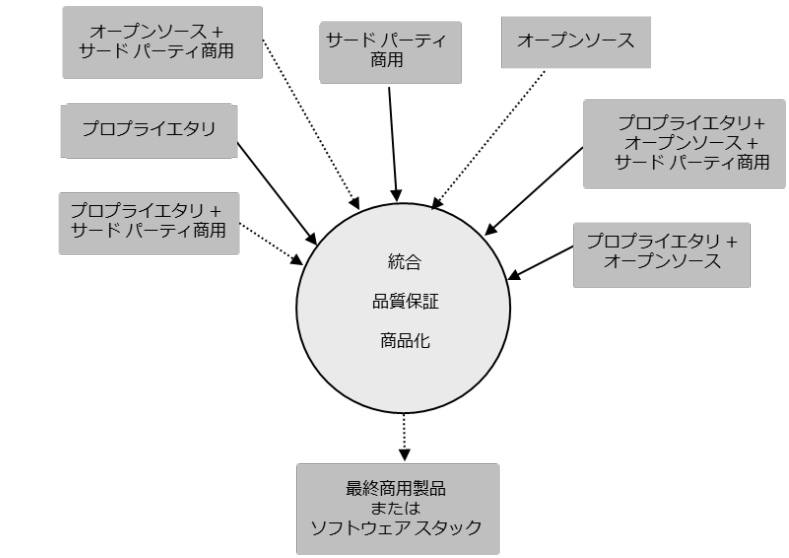


図 2：マルチソース開発モデル

かつては明らかにプロプライエタリなソフトウェア スタックだったものの中にオープン ソースのソフトウェアのコンポーネントが増えていくに従い、ビジネス環境は慣れ親しんだ領域、企業が好む環境から離れていきます。

図 3（次ページ）はあるプラットフォームやソフトウェア スタックにおいて、さまざまな階層でオープン ソース ソフトウェアを受けいれる様子を示したものです。

プロプライエタリな開発モデルと、マルチソースによる開発モデルとの大きな違いの一つは、オープン ソース ソフトウェアのライセンスは交渉するものではないことです。ソフトウェアの提供者（すなわちオープン ソースの開発者やプロジェクト）と調印する契約はありません。そうではなく、オープン ソース プロジェクトを開始した人々が所定のライセンスを選びます。そしてプロジェクトがある規模に達すると、変更は事実上不可能となります。マルチソースの開発モデルを使うとは、数十の相異なるライセンス（とライセンスの組み合わせ）に基づく数百人、時には数千人のライセンス提供者やコントリビューター（著作権者）と関わること、を企業は理解しなければなりません。その結果、かつては企業対企業のライセンスや合意に至る交渉を通じて管理されていたリスクは、強固なコンプライアンス プログラムと、注意深いエンジニアリングの実施によって管理されることになります。

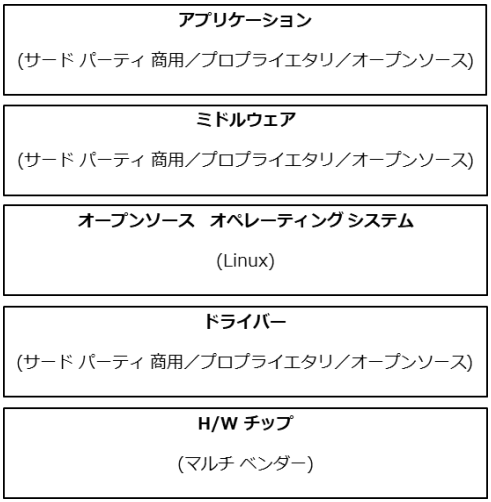


図 3：現代のソフトウェア プラットフォームの単純化したアーキテクチャー。個々のソフトウェアの構築ブロック内部でオープン ソースが増えていることを示している。

オープン ソースコン プライアンス手始め

オープン ソースのイニシアチブやプロジェクトを使う事で、企業や組織はオープン ソース ソフトウェアの開発者を代表する数百、時に数千のコミュニティと協力し、イノベーションを加速できるようになります。しかしながら、オープン ソースのコミュニティと組むには責任が伴います。オープン ソースのライセンスに伴う義務を順守することを保証しなければなりません。

オープン ソースのコンプライアンスとは、オープン ソースのユーザー、インテグレーター、開発者が著作権表示をきちんと見て、自らに関わるオープン ソースのソフトウェア コンポーネントのライセンス上の義務を満たすプロセスです。適切に規定されたオープン ソースのコンプライアンスのプロセスは、ライセンス条項の順守を保証すると同時に、企業自身、あるいはサードパーティのサプライヤの知的財産が意図せず公開されたり、その他不適当な結果を招いたりしないように設計されるべきです。

オープン ソースのコンプライアンスは 3 つの主要な目的の達成に役立ちます。

- ライセンス上の義務に従う
- オープン ソースが商用製品で効果的に使われることを助ける
- サードパーティとの契約上の義務に従う

オープン ソースのコンプライアンスを保証する利点

オープン ソースのコンプライアンスを達成する事にはいくつかの利点があります。確固たるコンプライアンス プログラムを持つ企業は技術的にも利益を得ることが多いです。規定に従っているソフトウェア群の資産はサービスし、試験し、アップグレードし、保守することが容易だからです。さらに、コンプライアンスの活動は、さまざまな製品や部門で使われていてカギとなる、組織にとって高度に戦略的で利益をもたらすオープン ソース ソフトウェアの洗い出しにつながります。また逆に、レビューを繰り返すことを通じてコンプライアンスは、オープン ソースのコンポーネントの利用に伴うコストやリスクを示すことにもなります。

健全なコンプライアンスのプログラムは外部コミュニティと協業する上でも大いに有用です。コンプライアンス上の問題が生じた時、そうしたプログラムは会社が善意であることを示すことができます。

最後に、頻度は下がるが強固なコンプライアンスのプログラムがもたらす利益として、例えば、会社の買収や売却、新製品や新サービスのリリースのためより良い準備となる、が挙げられます。オープン ソースへのコンプライアンスの保証はそうした業務が完了する前に終了させることが必須だからです。さらには、OEM や下流のベンダーとの取引においてコンプライアンスが検証できることは付加的な優位となります。

コンプライアンスの失敗

ソフトウェア開発の全工程において、コンプライアンス プログラム上のエラーや不十分さはコンプライアンス上の失敗に繋がることがあります。下記はこうした失敗の例です。

- **不適切な帰属表示。** 帰属表示は通常、テキストファイルとしてオープン ソースのコンポーネントと共に提供され、オープン ソース コンポーネントへのコントリビューターが提供したことを示します。
- **ライセンス表示提供を怠る。**ライセンス表示とは、製品やスタックに含まれるオープン ソースのライセンス文書を含んだファイルで、多くの場合、製品の文書と共に、あるいは製品やアプリケーションの UI と共に提供されます。
- **著作権表示の省略。**著作権表示は、ソフトウェアのコピーにつけられる識別情報で、著作権者を示します。
- **変更通知の付け忘れ。**変更通知とは、ソース コードに対する変更をチェンジログ内で表示する、GPL や LGPL で必要になるものです。一例を下記に示します。

```
/*
* Date         Author           Comment
* 10/15/2015  Ibrahim Haddad    Fixed memory leak in nextlst()
*/
```

- **製品ドキュメントや製品広告での不十分な、あるいは誤解を招く主張。**
- **ソース コード提供の失敗。**ソース コードを利用可能とすること（変更を含む）は GPL/LGPL ライセンス ファミリーの要求の一つです。

- **GPL/LGPL 等のライセンスのソース コードを使う際に書面で告知しない。**書面での告知は、その製品のエンド ユーザーに対してその製品に含まれるオープン ソース ソフトウェアの情報、そして頒布されるべきソース コードをダウンロードする方法を開示します。これは通常、製品添付の文書に含めるとともに、製品のユーザー インターフェイスからアクセスできるようにします。書面告知の基本的な例は下記のようになります。

  FooBar の本製品の中で使われているソフトウェアに関連する、FooBar 社が公開しているソース コードのコピーを得るには、
  http://opensource.foobar.com にアクセスするか、opensource@foobar.com にメールでリクエストを送るか、またはリクエストを通常の郵便でお送りください

  FooBar 株式会社
  オープン ソース プログラム室
  住所
  市、州、郵便番号
  国名

- **ビルド用スクリプトを提供しない。**コンパイルに必要なもの（GPL や LGPL ファミリーのライセンス毎に）。

知的財産権上の失敗

表 1 (次ページ) はソフトウェア開発の過程においてプロプライエタリな知財とオープン ソースの知財を誤って混ぜてしまい、ライセンス コンプライアンス問題となるありがちな例を示したものです。もっともありがちなのは、異なる、または両立しないライセンス（例：プロプライエタリ、サードパーティやオープン ソース）下のソース コードを混ぜてしまう、ことです。そうして混ぜてしまうと企業はプロプライエタリなソース コードをオープン ソース ライセンスで開示することを強制され、（おそらくは）高い価値を持つ知財の制御を失い、市場における差別化が弱くなってしまいます。

知財上の失敗は下記を招きます：

- 差し止め命令によりコンプライアンス上の懸念が解決されるまで製品が出荷できない
- 問題のバイナリ コードに関わるプロプライエタリなソース コードを（ケースに応じた）オープン ソース ライセンスで頒布するよう要求される
- コンプライアンス上の懸念を解消するため多大な工数が費やされる
- 顧客、ディストリビューター、サードパーティのソフトウェア供給者、そしてオープン ソース コミュニティを困惑させます(文体？)

表 1：知財上の失敗の例

問題の類型	発見する方法	回避方法
オープン ソースのコードをサードパーティのコードに挿入 開発プロセスの中で開発者がオープン ソースのコード (いわゆるスニペット) をプロプライエタリやサードパーティのコードに挿入して発生	オープン ソースのコードとの一致の可能性を求めてソース コードをスキャン	コンプライアンスやオープン ソース ライセンス、プロプライエタリやサードパーティのコードにオープン ソースのコードを含めることの意味について認識するためのトレーニングを提供 予期されないライセンスやコードのスニペットのため、定期的にプロジェクトの全ソース コードをスキャン オープン ソース ソフトウェアを使う際には製品のレポジトリにコミットする前に承認を必要とするようにします

問題の類型	発見する方法	回避方法
オープン ソースをプロプライエタリなソースコードのソフトウェアにリンク (またはその逆。C/C++特有の問題) 衝突(conflict)する、または互換性のない(incompatible)[03] ライセンスのソフトウェアコンポーネントをリンクした結果発生	異なるソフトウェアコンポーネント間のリンクを検出できる依存性追跡ツールを利用; 企業のオープン ソースポリシーで許容されているかを識別	企業のコンプライアンスポリシーに基づくリンクのシナリオについての訓練を提供 定期的に依存性追跡ツールを動作させ、全てのリンク関係を検証; 企業のポリシーで許容されていないあらゆる事柄にフラグを立てます
オープン ソースのコンポーネントにプロプライエタリなコードを含めます 開発者がプロプライエタリなソース コードをオープン ソース ソフトウェアにコピー/ペーストして発生	ソース コードをスキャンする。ツールでオープン ソースのコンポーネント由来ではないソース コードは識別できるので、監査のため各種のフラグをつけます	スタッフを訓練 ソース コードを定期的に検査 プロプライエタリなソースコードをオープン ソースのコンポーネントに含めるには承認を必要とするようにします

ライセンス コンプライアンス問題

ライセンス コンプライアンス問題は知財上の問題に比べ一般に損害は小さいです。自社のプロプライエタリなコードをオープン ソース ライセンスで開示するといった副作用がないからです。

ライセンス コンプライアンス上の失敗は下記のいずれか（か、その組み合わせ）を招き得ます。

- ソース コードが開示されるまで製品出荷を差し止める命令

- バージョンのミスマッチによりサポートやカスタマー サービス上の問題となる（サポート ホットラインに電話や email でソース コード開示について問い合わせがくる結果）
- 顧客やオープン ソースコミュニティにおいて困惑や悪い評判を招く

表 2 にソフトウェア開発プロセスでの最も一般的なライセンス コンプライアンス問題の例、および回避方法を示します。

表 2：ライセンス コンプライアンス問題の例と回避方法

問題の類型	回避方法
ライセンス上の義務の一環としてソース コードのパッケージを公開したり利用可能としたりし損ねる	詳細なコンプライアンス チェック リストに従い、製品やアプリケーション、ソフトウェア スタックの出荷時には全てのコンプライアンス上のアクション アイテムが完了したことを確認
出荷されたバイナリと異なるバージョンのソース コードを提供	検証作業をコンプライアンス プロセスに追加し、提供するソース コードのバージョンが、出荷される バイナリのバージョンと確かに正確に対応することを確認
出荷製品に組み込まれたオープン ソース ソフトウェアへの変更をリリース忘れ	部品表（BoM） 差分検出ツールを使いリリース毎のソフトウェア コンポーネント変更を特定 コンプライアンス プロセスにおいて、より新しいバージョンのソフトウェア コンポーネントを再導入 「diff の計算」を変更された(オープン ソースとするにふさわしい) ソース コードのチェック リストに加え 、製品に使われたオープン ソースのリリース前に行う

問題の類型	回避方法
変更されたオープン ソースのコードにマークを付け忘れる、変更の記述を入れ忘れる	ソース コードへのマーク付けをソース コードのリリース前のチェックリストに加え、ダウンロードしたオリジナルのコピーとの差分すべてに確実にマーク ソース コードのリリース前に検査 変更されたソース コードが適切にマークされていることを、コンプライアンス プロセス上のマイルストーンとする ソース コードの変更履歴をアップデートすることを、開発プロセスの一部とするようスタッフを教育

コンプライアンス プロセス上の失敗

コンプライアンス プロセス上の失敗は、オープン ソースのライセンス条件の侵害、例えばライセンス上の義務を満たさない、に繋がる。表 3 にソフトウェア開発で起きる最も一般的なコンプライアンス プロセス上の失敗のリスト 、および回避方法を示します。

表 3：コンプライアンス プロセス上の失敗の例

問題の類型	回避方法
開発者が企業内のオープン ソース委員会 (時にオープン ソース評価委員会と呼ばれる) オープン ソース ソフトウェアの利用申請をしない、期限までに行わない	コンプライアンス ポリシーとプロセスについての教育 定期的にソフトウェア プラットフォーム の全スキャンを行い、承認された型に該当しないオープン ソースを検出。もしビルドされたシステムに該当するコンプライアンス チケットがないオープン ソース コンポーネントが検出されたら、自動で新チケットを発行（企業が Bugzilla のようなツールに埋め込まれているワークフローを使ってソフトウェア コンポーネントを追跡していると仮定） 業績評価にコンプライアンスを含める。例えばコンプライアンス ポリシーに従わなかったらボーナス査定に直結 開発者がオープン ソースの利用申請を早期に、オープン ソースのコード の利用を決める前でも行うよう命令
オープン ソースの訓練を受けない	オープン ソースの訓練は従業員のキャリア開発計画の一部であり、業績評価プロセスの一部としてモニターされていることを念押し
ソース コードの監査を行わない	コンプライアンスのスタッフを適切に訓練 ソース コードを定期的にスキャン 繰り返される開発プロセスのマイルストーンに監査を確かに入れ込む 適切なレベルのスタッフを揃え、監査がスケジュール遅れとならないようにする
監査で発見された問題を解決しない	監査レポートが完結しない限りコンプライアンス チケットをクローズ させない。チケットのクローズは、関連するサブのタスクが存在しない時のみとする

学んだ教訓

この数年で明るみに出たコンプライアンス違反が数件ありました。そうしたコンプライアンス違反の法的な決着を通じて、オープン ソースのプロフェッショナル達は次のような教訓を学びつつあります。

製品出荷やサービス立ち上げに先立ちコンプライアンスを確実に行う

コンプライアンス違反事件の最も重要な教訓は、関係した企業は最終的には問題のライセンス条項に従わなければならなかった、のであり、この事実を踏まえた問題解決のコストは基本的なコンプライアンスのコストを圧倒的に上回っていた、のです。したがって、製品出荷やサービス立ち上げに先立ってコンプライアンスを確実に行うのが賢いやり方です。

コンプライアンスは法務部門の業務に留まるものではない、と認識することは重要です。全部門が関わって適切なコンプライアンス、正しいオープン ソースの利用、必要に応じた再頒布、を確実に行う必要があります。この関わりには首尾一貫したコンプライアンス ポリシーや手続きの確立と維持管理、利用中（プロプライエタリ、サードパーティ、オープン ソース）の全ソフトウェア コンポーネントのライセンスが共存し得ることを製品出荷やサービス立ち上げ前に確認すること、が含まれます。そのためには企業はオープン ソースを管理するインフラを末端まで構築し下記を行う必要があります。

- 製品の中で、サービスが開示している、あるいは内部で利用されている全てのオープン ソースを識別
- アーキテクチャーを評価し、オープン ソースのライセンス上の義務がプロプライエタリやサードパーティのソフトウェア コンポーネントまで拡大していないか、拡大しているならばどのように、を検証
- 適用可能なオープン ソース ライセンスを収集し、法務部門が評価
- オープン ソースの利用および頒布のポリシーと手続きを定める
- アーキテクチャー設計と製造の実務においてリスクを低減

コンプライアンス違反は高くつく

コンプライアンス違反が公となった事件の多くは GPL のソース コードが関わっています。これらの紛争解決の合意では下記の一つないし一つ以上 の条項が含まれています。

- コンプライアンスを満たすよう必要なアクションを取る
- コンプライアンス オフィサーを置きコンプライアンスをモニターし確認させます
- 製品を受け取った過去の顧客に、製品はオープン ソース ソフトウェアを含んでいること、そのソフトウェアについての権利、を通知
- ライセンス告知を企業のウェブサイトに掲載
- 製品説明に告知を追加
- ソース コードをすべての改変と共に利用可能 に（GPL/LGPL ファミリーのライセンスに限る）
- 問題となっているオープン ソース ソフトウェアのバイナリの頒布を、関連のソース コードが開示されるまで、あるいはコンプライアンス違反で影響を受けた特定顧客が利用可能となるまで差し止め. いくつかの事件では、原告に対し非開示の額の金銭的対価

さらに、コンプライアンス異議申し立てを受け、それが成功すると下記のコストを負うことになりました。

- コンプライアンスの照会への対応と発見と精査のコスト。企業は申し立てに応じて、調査とソース コードの精査を実施しなければなりません。
- 外部・内部のリーガルコスト
- ブランドや評判、信頼性へのダメージ

ほぼすべての事件において、オープン ソース ライセンス上の義務違反は面目の失墜、否定的な記事、オープン ソース コミュニティとの関係悪化を招いています。

関係は大事

オープン ソース ソフトウェアを自社製品で利用する企業は、利用しているコードを開発し維持七得るオープン ソース コミュニティと良い関係を持ち、維持することが望ましいです。オープン ソース プロジェクトのコミュニティは企業が自社製品に含まれるオープン ソース ソフトウェアのライセンスを尊重すると期待しています。その方向に動き、オープンかつ正直な関係を気づくことは大いに価値があります。

訓練は重要

訓練はコンプライアンス プログラムにおいて必須の構成要素であり、オープン ソース ソフトウェアの利用を律するポリシーを従業員がよく理解する礎となります。ソフトウェアに関わる全職員が企業のポリシーとプロセスを理解する必要があります。企業はそうした教育を公式・非公式の教育で提供することが多いです。

第 2 章

オープン ソース管理プログラムの確立

オープン ソース管理プログラムはオープン ソース ソフトウェアの全ての側面、その選択、承認、利用、頒布、監査、インベントリ、訓練、コミュニティとの関わり、広報等々の体系を定めます。本章はオープン ソース管理プログラムのさまざまな構成要素を概観し、新しいコンプライアンス プログラム策定における難しさを概説し、そうした難しさを克服する方法を述べます。

オープン ソース コンプライアンス プログラム

最初に、成功するオープン ソース コンプライアンス プログラムでコアとなる構成要素を概観します。本節と図 4 にこれら構成要素の概観を示すします。

図 4：オープン ソース 管理プログラムの必須構成要素

コンプライアンス戦略

オープン ソース コンプライアンス戦略は、ポリシーやプロセスの実装 の主たる側面について、ビジネスに立脚してコンセンサスを推し進めるものです。こうしたハイレベルなコンセンサスに基づかずにポリシーやプロセス の実装の細部について社内合意しようとすると、不可能とは言わないまでも非常な困難に直面します。コンプライアンス戦略はコンプライアンスを確保するために行うべきことを定め、職員のオープン ソース ソフトウェアの扱いを統べる原則を与えます。オープン ソースの承認、取得 , 利用、オープン ソース ライセンスを含むかオープン ソース ラインセンスに基づいてライセンスされたソフトウェアをリリースする方法、の正式なプロセスも本戦略に含まれます。

照会応答戦略

照会応答戦略は、コンプライアンスが問われた時に行うべきことを定めます。企業は時として悪評（場合によっては正式な申し立て）を、コンプライアンスについての追加情報提供の要請を無視したり、コンプライアンスの照会を扱う方法を知らなかったり、オープン ソース コンプライアンス プログラムが欠落していたり不十分だったり、単に照会者と協力しなかったために、受けてきました。こうしたアプローチは関係者のだれにとっても実りや利益を生みません。したがって企業は照会を受けとり、受領したことを応答し、照会者に検討することを伝え、現実的なフォローアップの日程を通知する方法を定めるべきです。後ろの章で、オープン ソース コンプライアンスに対する照会を扱うシンプルなプロセスについて述べます。

ポリシーとプロセス

オープン ソース コンプライアンス ポリシーとは、オープン ソース ソフトウェアの管理（利用と寄付の双方）を取り仕切るルール群です。プロセスとは、日々の業務にこれらルール群を実装するかを詳細に列挙したものです。コンプライアンス ポリシーとプロセスがオープン ソース ソフトウェアのさまざまな側面、利用、寄付、監査、頒布を取り仕切ります。

図 5（次ページ）はコンプライアンス プロセスの例を図示したもので、個々のソフトウェア コンポーネントが精査の一環として通るべきさまざまなステップを示しています。このプロセスの詳細は後ろの章で述べます。

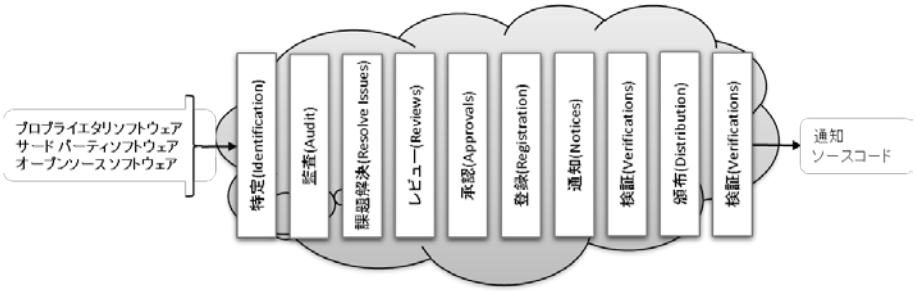


図 5 コンプライアンス精査プロセスの例

コンプライアンス チーム

オープン ソース コンプライアンス チームとは、オープン ソース コンプライアンスを確実に行うためさまざまな分野の人々を集めたチームです。コアとなるのはオープン ソース評価委員会 (Open Source Review Board : OSRB) と呼ばれることが多い委員会で、エンジニアリング、製品のチームの代表者たち、一人以上の法務顧問、そしてコンプライアンス オフィサーから構成されます。拡大チームはさまざまな部門からコンプライアンス活動のため随時加わる人々で、文書作成、サプライチェーン、全社開発、IT、ローカル化、オープン ソース幹部会議 (Open Source Executive Committee : OSEC)らが含まれます。ただし拡大チームのメンバーはコアチームと違い、OSRB から仕事を委任された時だけ活動します。コンプライアンスに常時携わるわけではないです。第 3 章ではオープン ソース コンプライアンスを達成するための個々の人間の役割と責任について詳述します。

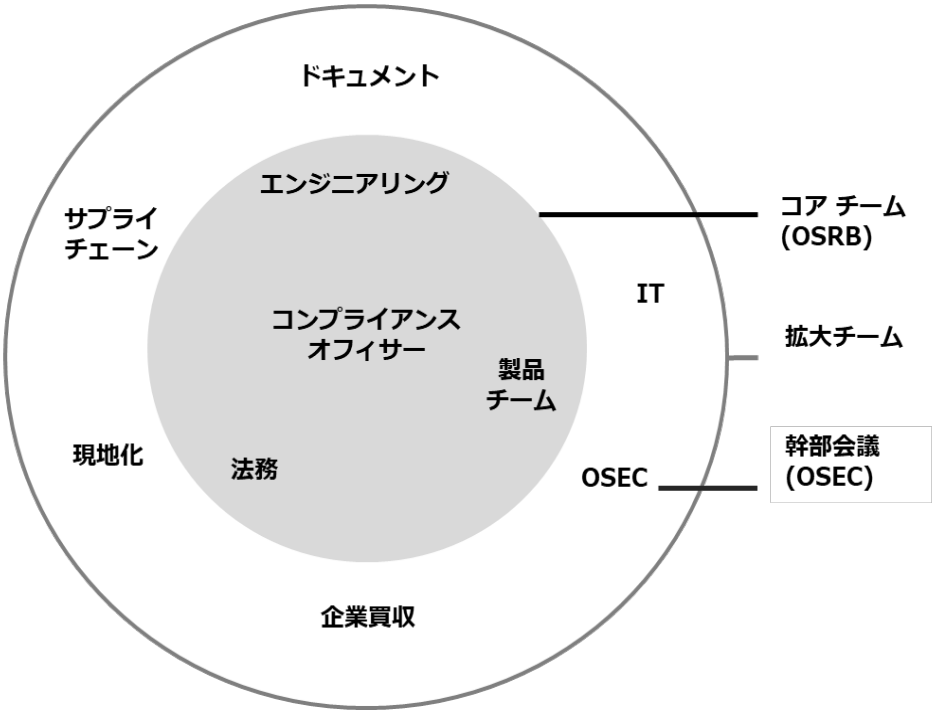


図 6 オープン ソース コンプライアンスを確かなものとする個人とチーム

ツール

オープン ソース コンプライアンス チームはツールを使うことでソース コードの監査、オープン ソース コードとそのライセンスの発見、が自動化され楽になります。こうしたツールには

- コンプライアンス プロジェクト管理ツール、コンプライアンス プロジェクトを管理し、タスクやリソースを追跡
- ソフトウェア インベントリ ツール、個々のソフトウェア コンポーネント、バージョン、それを使用している製品、関連情報を継続的に追跡

- ソース コードとライセンスの識別ツール、ビルドするシステムに含まれるソース コードの由来とライセンスを特定
- リンク解析ツール、問題となる C/C++ソフトウェア コンポーネントと、製品に含まれる他のソフトウェア コンポーネントとの関係を特定。本ツールを使うことで、企業のポリシーを満たさないソース コード パッケージとのリンクを発見できます。この目標は、プロプライエタリやサードパーティのソフトウェア コンポーネントにオープン ソース上の義務が何ら及んでいないことの確定です。もしリンクが発見されたら、製造部門に対しバグチケットを発行し、問題と解決方法を記述します。
- ソース コードのピアレビュー（査読）ツール、ライセンス上の義務を満たすための開示に先立ちオリジナルソース コードに対する変更をレビュー。
- 部品表 (bill of material, BOM) 差分ツール、二つの異なるビルドでの部品表の違いを特定。本ツールはコンプライアンス準拠作業を積み上げで行う場合に非常に有用です。

ウェブ上のプレゼンス

企業はポータルを 2 方向で使います：一つは内向き、企業内へ。もう一つは外向き、社会とオープン ソースコミュニティへの窓として。内部向けのポータルはコンプライアンス ポリシー、ガイドライン、各種文書、訓練、通知、メーリングリストへのアクセスを提供します。外部向けのポータルは社会とオープン ソースコミュニティに向けて公開のプラットフォームとなり、またオープン ソースのパッケージのソース コードや受領通知ほかの情報開示を投稿する場としてライセンス上の義務を履行します。

教育

教育はコンプライアンス プログラムにおいて必須の構成要素で、オープン ソース ソフトウェアの利用を取り仕切るポリシーを従業員がよく理解する基礎となります。オープン ソースとコンプライアンスの教育の目標は、公式、非公式を問わず、オープン ソースのポリシーや戦略、オープン ソースのライセンスや、製品やソフトウェア ポートフォリオにオープン ソース ソフトウェアに組み入れることのビジネス的、法的リスクについて共通理解を作り上げ、注意喚起することです。またトレーニング は、企業内でコンプライアンス ポリシーを広め推進し実施し、コンプライアンスの文化を育てる場となります。

公式トレーニング

企業の規模やオープン ソースの営利活動への浸透度合いに応じて、企業はオープン ソースに携わる従業員に公式の講師によるトレーニングコースを命じることができます。実例でのトレーニングがその頂点となりえます。

非公式トレーニング

非公式のトレーニングは下記のいずれか、あるいは全部を含みます。

- **ブラウンバッグ セミナー：**ブラウンバッグ（訳注：昼食が入った袋） セミナーとは一般に、昼食時に従業員や招待者が行うプレゼンテーションです。こうしたセミナーの目標は、商用の製品や企業のソフトウェア ポートフォリオの中にオープン ソースを組み込むことのさまざまな側面について講演し、議論を起こすことです。これらのセッションは、企業のコンプライアンス プログラム、ポリシー、そしてプロセスに関する議論も含みます。
- **新人研修：**時にはコンプライアンス オフィサーが企業のコンプライアンスの努力、規則、ポリシー、プロセスを新人に対して研修の一部として講演し、必要となるオープン ソースの管理上の情報（聞くべき人、訪れるべき内部のウェブサイト、オープン ソースとコンプライアンスのトレーニングへの参加方法等々）を提供します。

自動化

オープン ソース ソフトウェアを使いたい、寄付したい開発者はオンラインで申請し、適切な承認を得る必要です。これは自動化されたオンラインシステム、通常はオープン ソース コンプライアンスの管理を特に組み込んだワークフローに従うバグ追跡システムで行うのがベストです。

メッセージの発信

メッセージの発信は内部向け、外部向けともコンプライアンス プログラムの必須の構成要素です。ここで最も重要なのは明確で首尾一貫していることで、内部的に従業員に対してオープン ソース関連の会社の目標や懸念事項を伝える場合も、外部に向け自社製品やソフトウェア スタックが利用しているオープン ソース プロジェクトの開発者コミュニティに伝える場合も同様です。

コンプライアンスの難しさと解決策

企業でのコンプライアンス プログラムの確立では、ほぼ確実に困難に直面します。以下の節では最もありふれた困難を取り上げ、克服する方法を示唆します。

コンプライアンス プログラムの作成

最初の難しさは、コンプライアンス プログラムやそれを支える（既存の）内部手続きのためのインフラと、製品出荷やサービス開始のメ切とのバランスを取ることです。こうした難しさを和らげ解決し、開発活動にとって重荷とは見做されない能率的なプログラム確立を助けるさまざまな手段があります。

解決策

役員のサポート

役員レベルでのオープン ソース管理プログラムへのコミットは、その成功と継続のために重要です。

ポリシーとプロセスの負荷の軽さ

プロセスとポリシー は重要です。しかし開発プロセスにとって重荷に過ぎると開発チームが見做さないよう軽く効果的でなければなりません。

能率的なオープン ソース管理は二つの基盤の上に成り立ちます：簡単で明確なコンプライアンス ポリシーと、軽いコンプライアンス プロセスです。

基本ルールを強制

コンプライアンス プログラムを実施する一環として、誰でも従わねばならない幾つかのシンプルな規則を定める必要があります。

- 開発者がオープン ソース ソフトウェアを製品やソフトウェア スタックに組み入れを計画するときは申請します。
- サードパーティのソフトウェア サプライヤに、提供物に含まれるオープン ソース ソフトウェアの情報を開示させます。あなたのサプライヤがオープン ソース コンプライアンスを見事に実践しているとは限りません。オープン ソースの開示に関わる文言を入れるよう、契約を見直すべきです。
- アーキテクチャーのレビューとコードの検査を指示し、オープン ソース評価委員会 (OSRB) がソフトウェア コンポーネント間の関係を理解し、オープン ソースからプロプライエタリなソフトウェアへとライセンス上の義務が伝搬していることを見つけられるようにします。大規模に行うには適切なツールが必要となります。
- サードパーティのソフトウェア プロバイダーから受領するすべてのソフトウェアをスキャンし、オープン ソースに関する開示が正しく、完全であることを確認します。

コンプライアンスを開発プロセスに統合

コンプライアンス確立の最善の方法は、コンプライアンス プロセスやポリシー、チェックポイント、活動を既存のソフトウェア開発プロセスに組み込むことです。

長期的な目標と短期的な実行

図 4 はコンプライアンス プログラムの成功に必須の構成要素を示しています。プログラムのすべてを実装するのに必要な作業の多さに圧倒される関係者もいるかもしれません。実際にはそこまで難しいものではありません。すべての要素を同時に実装しなければならない訳ではないからです。すべての組織において優先されるのは、製品やサービスを期限通りに出荷し、と同時に内部的なオープン ソース コンプライアンスのインフラストラクチャー を構築し拡大することです。したがって、企業なりのコンプライアンスのインフラストラクチャーを状況の進展に合わせて構築すべきと考えればよく、その際には将来の活動や製品をふまえ拡張性を考えるべきです。カギとなるのは思慮深く現実的な計画です。

解決策

- 長期的な目標に合致する、十全なコンプライアンスのインフラストラクチャーを計画し、短期的な実行の必要性に合わせ、要素を一つずつ実装します。たとえば、もしオープン ソースを含む製品開発やサービス提供を始めたばかりであり、すでに動いているコンプライアンスのインフラストラクチャーが何もないならば、最も差し迫った懸念はコンプライアンス チームを作ることであり、プロセスとポリシー、ツールと自動化を確立することであり、従業員を訓練することです。これらの活動を（この順番で）開始し、システム構築を（コンプライアンスの観点から）適切に制御しつつ開始したならば、プログラムの他の要素に進むことができます。
- ポリシーとプロセスを確立
- 開発プロセスの一部としてコンプライアンスを組み込み

コンプライアンスについての対話

コンプライアンス活動の成功を確かなものとするには、対話は必須です。2 種類の対話、あなたの組織内での内部的なもの、あなたの製品で使われているオープン ソースのプロジェクトの開発者コミュニティとの外部に向けたもの、を考えることが重要です。

内部的な対話

企業はコンプライアンスについての対話が内部的に必要となります。オープン ソースを商業ソフトウェアのポートフォリオへの組み入れが何をもたらすのか従業員が理解していることを確実にするためであり、企業のコンプライアンス ポリシー、プロセス、ガイドラインについて教育を受けていることを確かなものとするためです。内部的な対話は下記のいずれでも可能です。

- オープン ソース コンプライアンス活動を幹部がサポートする電子メールでの対話
- オープン ソース ソフトウェア関連業務に携わる全従業員に対する正式な訓練命令
- オープン ソースとコンプライアンスについてのブラウンバッグ セミナーによりコンプライアンスについてさらに注意喚起し、活発な議論を奨励
- 内部的なポータルを作り、企業のコンプライアンス ポリシーやプロセス、オープン ソースに関係した出版物やプレゼンテーション、メーリングリスト、オープン ソースとコンプライアンスに関係する討論フォーラムをホスト
- 企業全体に向けたオープン ソースのニュースレター。一般に隔月か四半期ごとに、オープン ソースのコンプライアンスについて注意喚起

外部との対話

企業は外部とのコンプライアンスについての対話が必要となります。自社が製品で使うオープン ソース ソフトウェアについてライセンス上の義務を果たす努力をしていることを、オープン ソースのコミュニティに確かに認識させるためです。

外部との対話は下記のいずれでも可能です。

- コンプライアンス遵守のためにオープン ソースを頒布するためのウェブサイト
- オープン ソースの組織への参加 やサポート。こうした活動は企業がオープン ソースの組織との関係を確立し、そうした組織の役割を理解し、有意義な貢献をするのを助けます。
- オープン ソースのイベントや会議への参加。参加にはさまざまなレベルがある。イベントのスポンサーから講演や出版物への寄与、あるいは単に開発者を参加させオープン ソースの開発者たちと知り合いオープン ソース コミュニティのメンバーと新たな関係を育成します。

クリーンなソフトウェア起点 の確立

コンプライアンス プログラムの開始当初の困難の一つは、使われているオープン ソース ソフトウェアとそのライセンスを正確に把握することです。この初期の監査プロセスは、製品やソフトウェア ポートフォリオのクリーンなソフトウェア起点の確立、と呼ばれることが多いです。これは数か月かかることもある重い活動で、開発と並行してどの程度早期に作業を開始するかに掛かっています。

解決策

組織は初期のコンプライアンスを以下の活動を通じて達成します。

- オープン ソースの利用申請の早めの提出と評価
- 自動化されたソース コードのスキャン を継続的に、事前設定した時間間隔で全ソース コードに対して 実行
- ソース コードのベースを継続的にスキャン。これにはサードパーティのソフトウェア プロバイダーから受領したものを含めます。該当するコンプライアンス チケットなしにコードベースにチェックインしたソース コードを捕まえるためです。こうしたソース コードのスキャンはたとえば毎月行います。

- 設計とアーキテクチャーの評価をソース コードの検査に加えて強制します。これはオープン ソース、プロプライエタリ、そしてサードパーティのソフトウェア コンポーネントのコードの関係を解析するためです。これを強制するのは、そうした関係がライセンス上のコンプライアンスの義務に関わるときのみでよいです。

もし企業がコンプライアンスの起点構築に失敗すると、その製品の将来の改訂版（またはおなじ起点を使った異なる製品）がコンプライアンス問題で苦しむことはほぼ確定となります。

そうしたシナリオを防ぐため、企業は下記を考慮すべきです。

- シンプルだが確実に行われるポリシーと軽量なプロセスを提供
- コンプライアンス上のチェックポイントをソフトウェアの開発プロセスに含め、コンセプトが出荷される製品やソフトウェア スタックとなる時に行います。理想的にはすべての開発上のマイルストーンにおいて対応するコンプライアンス上のマイルストーンを組み込み、ビルドに使われる全ソフトウェア コンポーネントが対応する、承認されたコンプライアンス チケットを確実に持つようにします。
- 専任のコンプライアンス チームを確保します。この点は後ろの章で文章を割きます。
- コンプライアンス チケットを効率的に処理するためツールや自動化を利用します。この点は後ろの章で論じます。

コンプライアンスの維持

オープン ソースのコンプライアンスを維持するには、コンプライアンスの起点確立と同様、いくつかの困難があります。実際には踏むべきステップの多くは同じで、ただし規模は小さく、積み増しとなります。コンプライアンスの維持は継続的な努力であり、規律、そして既存の製造およびビジネスのプロセスへのコンプライアンス活動組み込みのコミットメントによります。

図 7 に積み上げのコンプライアンスの概念を示します。ここでは初期のコンプライアンスの起点と現在のバージョンとの間で起きたソース コードの変更が何であれ、コンプライアンスを確実に満たすことが必要となります。

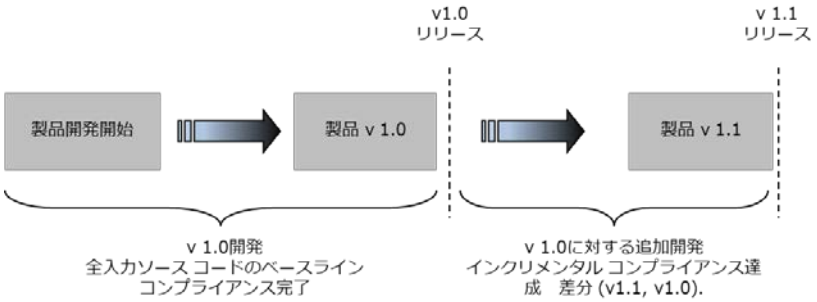


図 7：積み上げのコンプライアンスの例

解決策

企業は以下の活動を通じてコンプライアンスを維持することができます。

- 企業のコンプライアンス ポリシーとプロセス、加えてあらゆるガイドラインの厳守コードベースに統合されたすべてのソース コードについて、出所を問わず継続的に監査
- コンプライアンスや自動化に使われるツールを継続的に改善し、コンプライアンス プログラムにおいて可能な限り多くのプロセスを高い効率で実行

内面化と末永い実行

オープン ソースのコンプライアンス活動を維持するのは組織 が成長し、オープン ソースを使った更なる製品やサービス出荷に伴う進行中の挑戦です。企業は開発のカルチャーの中へとコンプライアンスを内面化し、末永く実行するためにいくつかの段階を踏むとよいです。

解決策

後援者

役員レベルのコミットメントがコンプライアンス活動を持続させる上で必須です。進行中のコンプライアンスのリーダーとなり、オープン ソースの管理機能を企業としてサポートする役員がいなければなりません。

一貫性

企業全体で一貫性を保つことは複数のビジネス ユニットや子会社を持つ大企業では重要です。部門間での一貫したアプローチは、記録の保持、グループ間でのコードの共有促進と並び有用です。

計測と分析

コンプライアンス活動やプロセス、手続きのインパクトや効率性を計測し分析すべきです。パフォーマンスを調べ、コンプライアンス プログラムを改善するためです。計測の尺度は、コンプライアンス プログラムを推進する場合にプログラムの個々の構成要素から得られる生産性向上について対話をする助けとなります。

コンプライアンス プロセスの改善

組織がオープン ソースを利用する範囲や目的 は変化します。製品や技術、企業の吸収や合併、海外への委託、その他多くの要因によって動きます。したがって、コンプライアンス ポリシーとプロセスを絶えず評価し、改善する必要があります。

さらに、オープン ソース ライセンスの解釈や法的リスクも進化します。こうしたダイナミックな環境の中で、コンプライアンス プログラムも進化する必要があります。

守らせること

コンプライアンス プログラムは、守られなければ無意味です。効果を持たせるため、コンプライアンス プログラムにはプログラム厳守を監視し、ポリシーやプロセスやガイドラインを全社的に守らせる メカニズムを含めるべきです。コンプライアンス プログラムを守らせる方法の一つは、ソフトウェア開発プロセスと統合し、従業員の業績査定の幾分かをコンプライアンス プログラム活動へのコミットメントと実行とすることです。

スタッフ割り当て

スタッフがコンプライアンスの職務のために割り当てられ、十分なコンプライアンスの訓練が組織の全従業員に確実に施されていなければなりません。大きな組織ではコンプライアンス オフィサー、および関連する職務はフルタイムの仕事となり得ます。小さい組織では分担し、時間の一部を割くべき活動となる可能性が高いです。

第 3 章

コンプライアンス達成に向けて：

役割と責任

一人の個人では、どれほどの達人であろうとも、オープン ソース コンプライアンスを組織全体に実装することはできません。図 8 はオープン ソース コンプライアンスの達成に責任があるさまざまな部門を書き下したものです。コンプライアンス達成には二つのチームが必要となる。コアチームと拡大チームです。後者は通常、前者を包括するものとなります。

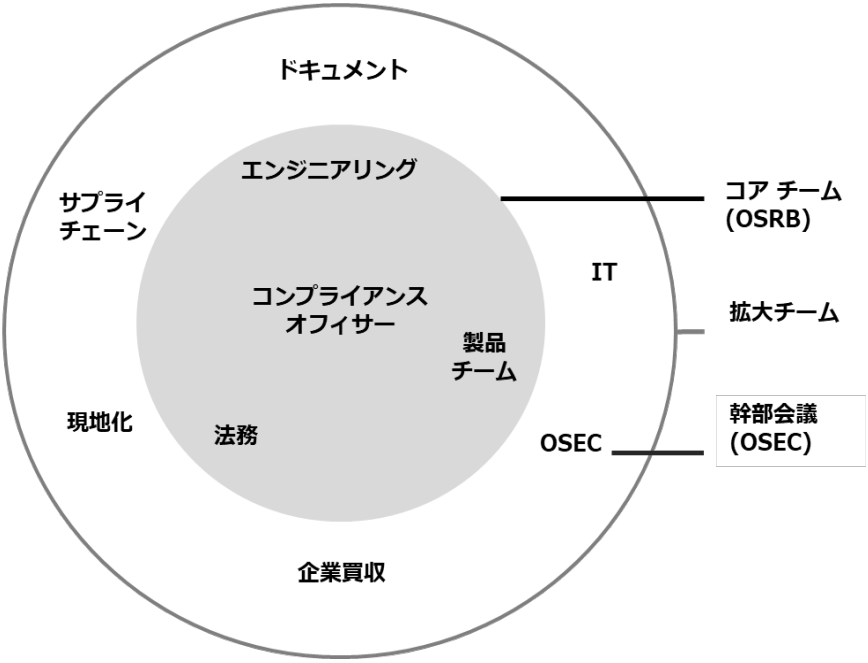


図 8：オープン ソース コンプライアンスを確かなものとする個人とチーム

コアチームはオープン ソース評価委員会 (Open Source Review Board : OSRB) と呼ばれることも多いです。エンジニアリングと製品のチームの代表者たち、一人以上の法務顧問、そしてコンプライアンス オフィサーから構成されます。表 4 に個々の参加者のコアチームにおける役割と責任を示します。

拡大チームは表 5 (49 ページ) に示す通り、さまざまな部門からコンプライアンス活動のため随時加わる人々で、文書作成、サプライチェーン、全社開発 、IT、ローカル化、オープン ソース幹部会議 (Open Source Executive Committee : OSEC) らが含まれます。ただし拡大チームのメンバーは（実体をもつ組織である）コアチームと違い、OSRB から仕事を委任された時だけ活動します。

表 4 コンプライアンス コアチーム (OSRB) の主な役割と責任

参加者	主な役割と責任
法務の代表者 代表者はその時の作業に応じ、法律顧問から弁護士補助職員まで変化します	OSRB と OSEC へ参加 オープン ソース ソフトウェアの利用、改変、頒布をレビューし承認 ライセンスについてのガイドラインの提供 トレーニングの提供と承認 オープン ソース コンプライアンス プログラム改善への貢献 オープン ソースのポータルの内容をレビューし承認 ライセンス上満たすべき義務一覧のレビューと承認 オープン ソースに関する告知の承認

参加者	主な役割と責任
エンジニアリングおよび製品チームの代表 企業によってはエンジニアリングと製品のチームは分かれていない	OSRB と OSEC へ参加 コンプライアンス ポリシーとプロセスの遵守 コンプライアンスの実践を開発プロセスの中に統合 コンプライアンス プログラムの改善に貢献 技術的なコンプライアンス プログラムの遵守 あらゆる質問への迅速な回答 設計、アーキテクチャー、コードのレビューの実施 頒布のためのソフトウェア パッケージ準備
コンプライアンス オフィサー オープン ソース コンプライアンス オフィサーは必ずしも専任である必要はないです。大抵の場合、オープン ソース部門の部長や所長が役割を果たします	全てのコンプライアンス活動を推進 ソース コードのスキャンと監査を調整 ソース コード パッケージの頒布を調整 OSRB と OSEC へ参加 コンプライアンスとオープン ソースのトレーニングに寄与 コンプライアンス プログラムの改善に寄与 OSEC にコンプライアンス活動について報告 自動化や、開発環境の中にオープン ソースのコードを見つける新たなツール作成に寄与

表 5 コンプライアンス拡大チームの役割と責任

参加者	主な役割と責任
オープン ソースの戦略を決定するオープン ソース幹部会議 (Open Source Executive Committee : OSEC)	知財のリリース提案 をレビューし承認 プロプライエタリなコードをオープン ソース ライセンス下でリリースする提案をレビューし承認これはそのソース コードはオープン ソースにする前提で開発されたならば不要です。
文書作成	オープン ソース ライセンスの情報と告知を製品の文書の中に含めます
ローカル化	製品やソフトウェア スタックに関わるオープン ソースについて、基本的情報をターゲットとなる言語に翻訳
サプライチェーン	サードパーティ プロバイダーからライセンスされた、または購入したソフトウェア コンポーネントに含まれるオープン ソースの情報の開示をサードパーティ プロバイダーに指示 オープン ソース ソフトウェアが含まれている、またはバンドルされているサードパーティのソフトウェアの納入を手助け
IT (情報技術)	コンプライアンス プログラムで利用されるツールと自動化のインフラをサポートし整備 OSRB の要請に応じ、新しいツールを作成、入手

参加者	主な役割と責任
企業買収	会社の合併や吸収買収に先立ちオープン ソース コンプライアンスが完了するよう要請 外部委託した開発センター、またはサードパーティのソフトウェア ベンダーからソース コードを受領するに先立ちオープン ソース コンプライアンスが完了するよう要請

オープン ソース評価委員会 (Open Source Review Board : OSRB)

OSRB は下記に責任を持ちます。

- サードパーティのソフトウェアとオープン ソースのソフトウェア ライセンスとが相互にコンプライアンスを確かに満たすようにします
- オープン ソース ソフトウェアの効果的な利用、およびオープン ソース ソフトウェアへの寄与を促進
- プロプライエタリな知財（引いては製品の差異化）を、オープン ソースのライセンスがプロプライエタリな、またはサードパーティのソフトウェアに確かに及ばないようにすることで防御

日々の活動としては、OSRB のメンバーは下記に従事します。

- エンド ツー エンド（端から端まで）のコンプライアンス プロセスを確立します。OSRB はエンドツーエンドのコンプライアンス プロセス、すなわち利用、監査、開発、関与、保障、コンプライアンス管理に責任を持ちます。エンドツーエンドのコンプライアンス プログラムについては 4 章で概観します。
- コンプライアンス プログラムで使われるコンプライアンス ポリシー、プロセス、ガイドライン、テンプレート、フォームを作成し保守

- オープン ソースの利用、改変、頒布の申請をレビューします。OSRB はエンジニアリングや製品部門からのオープン ソース利用申請 をレビューし承認します。利用のプロセスは 6 章で論じます。
- ソフトウェアの監査の実施：OSRB は製品に含まれる全ソフトウェアについて監査を行います。これは下記が含まれます。
- ソフトウェアベースに対しソース コードをスキャンするツールを適用
- スキャンするツールの結果を分析
- スキャンのツールで発見されたコード合致 、潜在的な合致、ライセンスの衝突を識別
- スキャンのツールで特定されたすべての問題の解決を監督
- 最終監査報告書を作成し、特定されたすべての問題が解決されたことを確実にする

監査は OSRB か独立した監査チームかが組織の規模に応じて責任を持ち、コンプライアンス オフィサーに報告します。6 章で監査のプロセスについて論じます。

- アーキテクチャーのレビュー実行。承認過程の一つとして、OSRB はエンジニアリング部門の代表と共にアーキテクチャーをレビューし、オープン ソースのソース コード、プロプライエタリのコード 、サードパーティのソース コードの関係を解析します。本レビューの目的は、アーキテクチャーのガイドラインが尊重され、オープン ソース、プロプライエタリ、サードパーティのソフトウェアの関係が、現場が受け入れ可能な法的ガイドラインの範囲内に収まっていることを確認することです。

- リンク解析をレビュー。OSRB は API 呼び出し等によるリンク関係により、何らかのオープン ソースのライセンス上の義務がプロプライエタリやサードパーティのソフトウェアに広がっていないかの確認のためリンク解析を行います。

- オープン ソースを含む製品のリリースやサービスの開始を取りやめる決定を検証

- 社内のスタッフやエンジニアから寄せられるオープン ソースに関する質問に対しガイダンスを提供

- 頒布前の検証作業の一環としてコードの検査を実施。オープン ソース ライセンスのテキストや著作権表示が完全な形となっており、ソース コードに対して施された変更を反映するようエンジニアたちが変更ログをアップデートしていることを確認

- 問題となっているオープン ソース ソフトウェアで満たすべきライセンス上の義務の一覧を作り、満たすよう関係部門に展開：OSRB がオープン ソースを製品に利用することを承認したら、承認プロセスの一部として OSDB は義務一覧を作り、関係するさまざまな個人やチームに展開して確実に満たすようにします。頒布前のプロセスの一部として OSDB は製品やサービスのリリース前に最終チェックを行いますが、義務を満たしていることの検証もここに含みます。

- オープン ソースとコンプライアンスの訓練を開発し提供。OSRB はオープン ソースとコンプライアンスの訓練の開発を主導し、従業員が企業のオープン ソースのポリシーやコンプライアンスのプロセスについて良く理解するようにします。さらに、OSRB は最も広く利用されるいくつかのオープン ソースのライセンス、商用でのオープン ソース利用に関わる話題についても教育すべきです。この訓練はオープン ソースを使ったソフトウェアの開発や管理に関わる従業員全員が受けなければなりません。

- 企業のオープン ソースのウェブサイトをホストし維持。：従業員向けの内部のウェブサイトはオープン ソースのプロセスやポリシー 、ガイドライン、訓練、告知が主となります。外部向けのサイトは通常、ソース コードのパッケージを利用可能とし、特定のコンプライアンス上の義務を満たすことが第一の目的となります。

- コンプライアンスの質問を取扱う。OSRB は企業に送付されたオープン ソースのコンプライアンスに関わるあらゆる質問に答える責任があります。第 9 章でコンプライアンスの質問を扱うプロセスを述べます。

- コンプライアンスの記録を取る：いかなるオープン ソース ソフトウェア コンポーネントについてもコンプライアンスに関するすべての記録が最新のものとなっていることに OSDB は責任を持ちます。

- エンドユーザー向け文書をレビューし、製品やソフトウェア スタックに含まれるオープン ソースについて、消費者へのコピーライト、帰属、ライセンスの告知が適切であることを確認します。また GPL/LGPL ファミリーのライセンスの場合には、可能であれば ソース コードを得る方法を書面にします。

- コンプライアンスの作業をより自動化し効率的にする、コンプライアンスの基盤の一部として利用されるべき新しいツールを推薦。

- 製品出荷をオープン ソース コンプライアンスの観点から承認。

- コミュニティに関与するポリシー、プロセス、手続き、ガイドラインを策定。これはコンプライアンスではないが、責任ある事柄のリストの完全性という観点で挙げておく。

法務

法律顧問はオープン ソース ライセンスへのコンプライアンスを確かなものとする委員会 、OSDB のコアメンバーです。法律顧問は 4 つの必須の義務があります。

1. 製品やサービスでのオープン ソースの利用を承認

オープン ソースを商用製品に用いる時には法律顧問の承認が必要です。典型的には、法律顧問はコンプライアンス チケットを評価します。これはオンライン トラッキング システム (例えば JIRA や Bugzilla) 、ソース コードをスキャンするツールの結果報告、ソース コードのパッケージに添付のライセンス情報を使います。

次にリスク要因を、エンジニアとオープン ソース コンプライアンス オフィサーからの情報に基づいて評価します。この作業の一環として、法律顧問は対象ソフトウェアコンポーネントについて入ってくる、および出ていくライセンスについて決定を行います。入ってくるライセンスとは、対象となるコードに含まれるソース コード全てのライセンスのこと です。出ていくライセンスとは、製品やサービスの 受領者が利用できるソース コードやオブジェクト ファイルのライセンスのことです。

2. オープン ソースのライセンスについて助言

- オープン ソース ライセンス上の義務についてガイドラインを提供
- 互換性がなく、衝突するライセンスに由来するライセンスの衝突について第一に企業のポリシーに基づいて助言を行います。これは時として関連するオープン ソースの組織の法律上の意見表明になど外部条件に依ることもあります。
- オープン ソースの利用に伴う知財権について助言します。これは特に、企業が元々はプロプライエタリだったソース コードをオープン ソース ライセンス下で提供するときに問題となります。
- エンジニアリング チームからのオープン ソースについての質問や懸念に対し勧告したり、ガイダンスを提供したりします。

3. エンドユーザー向け文書のアップデートをレビューし承認

こうした法務上のサポートは、製品に含まれるすべてのオープン ソースについて、適切なオープン ソースの告知 (著作権、帰属、ライセンス告知)を確実に消費者に提供するために行います。これに加え、GPL/LGPL ファミリーのライセンスのいずれかに従ってソース コードがライセンスされるならば、ソース コードを得る方法の情報を書面で提供する必要があります。

4. オープン ソース コンプライアンス プログラムの立ち上げと継続に貢献

- オープン ソースのポリシーとプロセスを確立し維持
- オープン ソースのコンプライアンスに関連して企業に送付された質問を取り扱う
- オープン ソースのライセンス、企業のポリシー、ガイドラインに関わる訓練を提供

エンジニアリングおよび製品チーム

エンジニアリングと製品のチームは OSRB に一人または複数の代表を出し、エンジニアリング チームに割り当てられたコンプライアンス関連のタスク全てを追跡し、適切、確実に解決します。これと並行し、エンジニアリングと製品のチームはオープン ソースのコンプライアンスについていくつかの責任を負います。

- オープン ソース ソフトウェアについて要望を提出：エンジニアリングと製品のチームは製品の基盤として導入すべき外部のソフトウェアを決定します。これはサードパーティやオープン ソースのソフトウェアを含みます。コンプライアンスの観点からのチームの主な責任は、製品やサービスに含まれる予定の全オープン ソースについて利用申請書類を出すことです。この書類には問題となるオープン ソースの利用方法を記述します。これはソフトウェアの起源と由来について良好な記録を構築し維持する助けとなります。
- 技術上のコンプライアンスのガイドラインの順守：エンジニアリングと製品のチームはソース コードに関する OSRB の構想、設計、統合、実装についての技術ガイドラインを遵守すべきです。OSRB のガイドラインは通常以下をカバーします。
  - よくあるミスと回避方法
  - リンクにより起き得る問題を回避するため、ライブラリやその他ミドルウェアを統合するときの規則

    - カーネル空間とユーザー空間のどちらで開発を行うか（Linux の場合）、特に組み込み環境においてプラットフォーム全体を開発する時
    - オープン ソース コンプライアンスに関わり得るエンジニアリング上の特別な状況
- 設計のレビュー：エンジニアリング チームは継続的に設計をレビューし、コンプライアンス問題をタイムリーに発見し救済策を講じるべきです。コンプライアンス オフィサーは設計のレビューを主導し、対象となるソフトウェア コンポーネントに応じて異なる参加者をエンジニアリング チームから招きます。
- OSRB との協力：エンジニアリング チームは OSRB からの質問に迅速に回答し、コンプライアンス チケットを解決するのに協力しなければなりません。
- 変更の追跡：変更ログを変更したオープン ソース コンポーネント毎に維持管理： オープン ソースのライセンス上の義務を満たすため変更ログが必要となります。問題となるライセンスに依りますが、ある種のライセンス（GPL/LGPL ファミリーなど）では変更されたファイルには、変更したこととその日付を目立つように付けることを求めています。
- 頒布のためのソース コード パッケージを準備：エンジニアリング チームはライセンスの義務に従って公開のウェブサイトで利用可能とするソース コード パッケージを準備します（これ以外のソース コードの配布方法については後ろの章で論じます）
- コンプライアンス上のマイルストーンを開発プロセスの一部に統合：これは OSRB とコンプライアンス オフィサーと協力して行います。
- オープン ソースの訓練を受ける：すべてのエンジニアは利用できるオープン ソースの訓練を受けねばなりません。
- オープン ソースのプロジェクトをモニターし、バグフィックスやセキュリティ パッチが利用可能となったかを判断し、製品で使われているオープン ソース コンポーネントのアップデートに責任を持ちます。個々のパッケージの組織内でのオーナーが通常はこの仕事を行います。

コンプライアンス オフィサー

コンプライアンス オフィサーは OSRB 議長やオープン ソース部長/役員とも呼ばれます。OSRB の議長でありコンプライアンス プログラムを管理します。

理想としてはコンプライアンス オフィサーは下記を出来る限り備えている必要があります。

- 法律顧問の業界慣行の知識と討議できる、一般的なオープン ソース ライセンスと義務について確たる理解
- 企業全体のポリシーとプロセスを確立できる知識と経験
- 企業の製品についての技術的知識オープン ソースについての歴史的な視点
- コミュニティのコンセンサスと慣行の知識
- 重要なオープン ソース プロジェクトのコミュニティとの人脈
- The Linux Foundation, Apache Foundation, Mozilla Foundation, Software Freedom Law Center (ソフトウェアの自由のための法律センター) などのオープン ソースの団体との人脈

OSRB に帰属する責任に加え、コンプライアンス オフィサーは以下の義務がある。

- コンプライアンスのエンドツーエンド (端から端まで) の精査プロセスを主導し、コンプライアンス プログラムの管理者として、コンプライアンスに関連する全ての業務が適切に認識され製品出荷を妨げるコンプライアンス上の問題が存在しないようにする
- ソース コードのスキャン作業を調整し、すべての監査を終わりまで主導する
- エンジニアリング チームによる設計のレビュー、コードの検査、頒布可能化の評価に参加し、エンジニアリングおよび製品チームがコンプライアンスの全プロセスとポリシーに従っていること、OSRB 承認済の利用申請書類に適合することを確認します。

- オープン ソースのパッケージのソース コード頒布（ライセンスに規定されている場合）についてエンジニアリングおよび製品チームと調整します。これは個々のオープン ソース パッケージの頒布チェックリストの用意と確認を含みます。
- OSEC と ORSB の間の連絡役となります。
- コンプライアンス上の課題を OSEC に上げます。
- オープン ソースの利用申請書類の承認について、エンジニアリングおよび製品チームと OSRB、OSEC の連絡役となります。
- コンプライアンス活動について OSEC に報告します。これは製品やサービスの出荷を妨げている課題の注意喚起を含みます。

オープン ソース幹部会議 (OSEC)

オープン ソース幹部会議 (OSEC) はエンジニアリング、法務、マーケティングの幹部に加えコンプライアンス オフィサーから構成されます。OSEC の責任はオープン ソース戦略の設定、知財の評価と解放の承認、従来はプロプライエタリだったソース コードを特定のオープン ソース ライセンス下で発表することの承認です。

文書化

文書化チームは、ソース コードを得る方法についての 書面での告知や、製品に添付するオープン ソースの通告すべてについて責任を持ちます。次ページの図 9 にそうした告知を準備し承認するプロセスを図示しています。このプロセスはコンプライアンス オフィサーが開始します。コンプライアンス オフィサーは製品が出荷されたら利用可能となる告知の書面や通告の草案を準備します。次に法律顧問が草案をレビューし、修正し、最終版を文書化チームに渡します。最後に、最終稿 を製品添付の文書に含めます。

コンプライアンス オフィサー	最終ソフトウェアのBOMに基づいて通知文書案と、通知を提示するべき場所と方法に関する提案を準備する
法務顧問	コンプライアンス オフィサーから出された提案をレビュー、編集し、承認を行う
ドキュメント チーム	法務顧問の承認に従って製品ドキュメントを更新する

図 9：製品添付の文書を更新し製品に含まれるオープン ソースの存在を反映する文書化チームの役割

ローカル化

ローカル化チームは製品に含まれるオープン ソース ソフトウェアの入手についてユーザーに知らせ、英語での適切な通告が利用可能となる ようにします。

サプライチェーン

サプライチェーン (ソフトウェアの調達) 部門 の手続きはオープン ソースの取得や利用を書き下したものに更新しなければなりません。サードパーティから供給されたソフトウェアは自分で検査するようにすることが強く望まれます。供給元から自社へのソフトウェアの移管には通常、サプライ チェーン関係者が関わります。サプライ チェーン部門 はオープン ソースのコンプライアンス活動をサードパーティのソフトウェア（そしてハードウェア）のサプライヤに対し、供給した品に含まれる全オープン ソースを開示するよう命令することで、またオープン ソースのパッケージとバンドルされたり統合されたりしているサードパーティのソフトウェアのライセンスを助けます。

この分野での最も良い方法は、サードパーティのソフトウェアプロバイダーに彼らが提供する品についてのオープン ソースすべてを開示するよう命令し、またオープン ソース上の義務を満たす計画を表明させることです。もしサードパーティのソフトウェアがオープン ソースを含んでいたならば、サプライチェーン部門 はオープン ソース ライセンス上の義務が確実に満たされるようにしなければなりません。受け入れた後はオープン ソースを含む製品やサービスを配送する業者として義務は自社のものとなるからです。自社のコードの受け手に対し、自社の「上流」の供給元を名指しし、ライセンス上の義務を果たす責任は供給元であり自社ではない、ということは受け入れられません。

IT

IT 部門はコンプライアンス プログラムで使われるツールのサポートや、自動化の基盤を提供、維持します。これは各種のツールやメーリングリスト、ウェブのポータルの提供が含まれます。さらに IT 部門は OSRB からコンプライアンス活動の効率向上のためのツールの開発や調達を要請されることもあります。

企業買収

企業買収部門はオープン ソースのコンプライアンスに主に二つの流れで関わります。買収と吸収合併、および開発の外部委託です

合併と買収

他社との合併や、自社が買収されることを考えるならば、コンプライアンス プログラムはしかるべきレベルの開示を行い、説明が出来るように作られるべきです。合併や買収についての全社ポリシーはオープン ソースを考慮するように改める必要があります。企業買収部門は合併や買収に先立ちソース コードをコンプライアンスの観点から評価し、議論を脱線させたり企業価値を変えたりする不測の事態が起きないよう命令しなければなりません。買収側の企業にとって、包括的なコードの評価はソフトウェア資産を正確に評価し、将来の企業価値が予期せぬライセンス問題にさらされるリスクを緩和することに繋がります。これに加え、買収側の企業は買収の合意文書にオープン ソースについての開示の条項を入れることも考えられます。精査の実務はオープ

ン ソースについての開示、開示されたオープン ソースとライセンスの評価のガイドを含むよう改める必要があります。

開発の外部委託

ソフトウェア開発の外部委託の合意文書も、コンプライアンス手続きを反映し、またその他の条項（表示や保証など）がオープン ソースによるリスクを広く含むよう改める必要があります。企業買収部門は外部の開発センターから受領したソース コード全てがコンプライアンス プロセスを通り、利用されたオープン ソースが全て発見されライセンス上の義務を満たすため適切な行動が確かに取られるよう命令しなければなりません。

その他の企業活動

企業買収部門はスピンオフやジョイント ベンチャーのコンプライアンスにも関わります。時にはコンプライアンス上の精査の結果、コンプライアンスの状況が理想からかけ離れており、ためにその活動を止める決定をする場合もあります。

第 4 章

オープン ソース コンプライアンス プロセス

オープン ソース コンプライアンス プロセスの実現は、コンプライアンスを組み込む必要がある開発プロセス、コードのサイズと数、関連する製品とサービスの数、外部から導入されたコード量、組織のサイズや体制など、多くの要因に基づいて、組織ごとに変化します。しかし、コンプライアンスの中心的要素は、通常同じです：コード中のオープン ソースを特定し、レビューし、利用を許可し、義務を順守することです。この章では、コンプライアンス プロセスの中心的要素に焦点を当てます。コンプライアンス精査の結果は、外部出荷を意図した製品において使われる全てのフリー オープン ソース ソフトウェアの特定と、それに付随するライセンス義務を遵守するための計画です。図 10 は、一貫したコンプライアンス プロセスの高水準の概要図の一例を提示しており、外部出荷される製品やサービスへの統合許可を受ける前にフリー オープン ソース ソフトウェアを含むコンポーネントが通過するさまざまなコンプライアンス ステップやフェーズを図示しています。

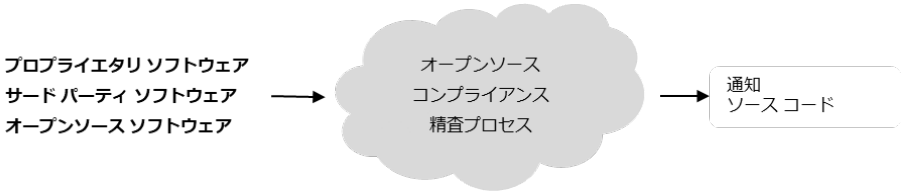


図 10：一貫したコンプライアンス プロセスの簡略図

図 10 に示されているのは一例であり、同じ目的を達成するためにコンプライアンス プロセスを作る方法は多くあります。この章を通じて、これらのさまざまなフェーズ、各フェーズのインプットとアウトプット、そしてコンプライアンス プロセスによってソフトウェア利用をどのように管理するかを確認していきます。

効果的なコンプライアンス

精査（due diligence）という用語は、ソース コード検査、ソース コード監視、品質義務やシステム監査の実行などを含む多くの概念を指しています。オープン ソース コンプライアンスのこのケースでは、精査は、以下の項目の確認を要求されるものとします。

- 製品で利用されるオープン ソース ソフトウェアは、特定され、レビューされ、承認されている。
- 製品実装は、承認されたオープン ソース コンポーネントとライセンスのみを含む。
- ライセンスされたものの利用に関する全ての義務は、特定されている。
- 適切な通知は、帰属表記や著作権表記を含む文書で提示されている。
- 変更（適用可能な場合）を含むソース コードは準備されて、製品出荷の時点で入手可能になっている。
- プロセス中の全段階の検証

簡潔で、組織内で十分に理解されている一貫したコンプライアンス プロセスを持つことには大きな利点があります。そのようなプロセスは：

- 組織に義務を履行させつつ、一方で組織がオープン ソースから恩恵を得られるようにします。
- オープン ソース利用を一時的なものから標準プロセスへと移行させます。
- オープン ソース コンポーネント入手の管理を助けます。
- 従業員が、責任ある方法でオープン ソースとどのように取り組めばよいか理解するのを助けます。
- 組織で利用しているさまざまなオープン ソース プロジェクトの開発者との関係を改善します。

- ソース コード変更の共有を行うことで、統合コードのプロジェクト コミュニティとの間で、情報やアイディアの交換が加速されます。
- 組織は安全にオープン ソース コンポーネントを採用して、新しいサービスや製品の立ち上げに必須のものとして利用できるので、イノベーションを加速できます。

一貫したコンプライアンス プロセスの要素

一貫したコンプライアンス プロセスの 10 ステップ

1. 入るソース コードの特定
2. ソース コードの監査
3. 監査で検出された課題の解決
4. 適切なレビューの完了
5. オープン ソース利用の承認受領
6. ソフトウェア リストへのオープン ソースの登録
7. オープン ソース利用を反映した製品文書の更新
8. 頒布前の上記全段階の検証
9. ソース コード パッケージの頒布
10. 頒布に関連した最終検証の実施

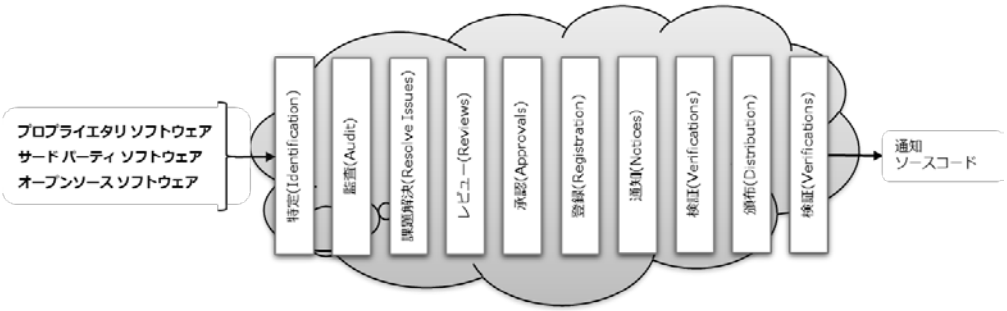


図 11：一貫したコンプライアンス プロセス

この章の残りの部分では、これらの 10 ステップについて詳しく説明します。

ステップ 1 — オープン ソースの特定

この最初のステップの目的は、標準パッケージや、サード パーティか自社開発ソフトウェアに組み込まれた形で、オープン ソースがソフトウェア ポートフォリオに入り込んだり統合されたりすることを監視することです。製品に利用されるオープン ソースを特定するいくつかの方法があります。

- オープン ソース利用の要求：これは、製品中のオープン ソース利用を特定する最も普通の方法です。技術担当か製品マネージャは、OSRB（3 章で説明）かコンプライアンス チームに特定製品やプラットフォーム リリースに特定のオープン ソースを利用したい旨を通知することが求められています。申請者は、レビューと承認の対象であるオープン ソース パッケージの計画した利用に関して情報を提供します。
- コンプライアンス基盤として確立するために、全プラットフォームや製品コード ベースを監査し、その上で、後続のリリースで変更されたコード モジュールを監査する。

- サード パーティ ソフトウェア プロバイダー精査：これは、オープン ソース コンプライアンス チームによる開示内容のレビューを伴う、サード パーティ供給者によって提供された製品中にあるオープン ソース コンポーネントの全開示を要求します。あるケースでは、精査の追加として、供給コードの監査結果をサード パーティ ソフトウェア ベンダーに要求することも意味があります。これは、製品へ入るオープン ソースの管理を確実にします。
- プロプライエタリ（自社開発の） ソフトウェア コンポーネントの監査：例として、技術者が、オープン ソース コンポーネントにあるソース コードをコピー／ペーストして、プロプライエタリ ソフトウェアに加えるかもしれません。それゆえに、製品出荷日前に発見されなければコンプライアンス違反に陥るようなオープン ソース コードを含んでいるかもしれませんので、自社開発ソフトウェア コンポーネントを監査することは重要です。

組織のソース コード リポジトリに入る、オープン ソースの利用要求に対応していない全オープン ソース コンポーネントの検査：オープン ソース利用の意図のフォームへの記載を技術者に頼るのは、全ての入るオープン ソース ソフトウェアを説明するには必ずしも確実な方法とはいえません。それゆえに、バックアップとして、オープン ソース用の分離したフォルダーを持つソース コード管理システムの準備と、このフォルダーにチェックインがされた時の警告を検討して下さい。オープン ソースと自社開発プロプライエタリ ソフトウェアとサード パーティ ソフトウェアをビルド システムで別のフォルダーに分離しておくのは常に奨励されるプラクティスですから、新しいコードが提出された時に警告が出るようにすることはできます。既存の利用（オープン ソース要求）フォームに対応しない新しいコンポーネントが提出されたら、それは新しいコンポーネントであり、新しいフォームに記入する必要があります。

特定フェーズの前提条件

以下条件の一つが満たされる：

- . 特定のオープン ソース利用を要求する入る OSRB フォーム
- プラットフォーム スキャンによって（適切な承認なしに）利用されているオープン ソースの発覚

- サード パーティ ソフトウェアの一部として利用されているオープン ソースの発覚

特定フェーズの結果

- コンプライアンス記録は、オープン ソースに対して生成（更新）されます。
- ソース コードをスキャンするように、監査が要求されます。

ステップ 2 — ソース コードの監査

コンプライアンス精査の 2 番目のステップは、既知のオープン ソース プロジェクトとの一致を検出する自動解析ツールを使ったスキャンで構成されます。

監査担当者は、リリースに含まれているものはさまざまな適用可能なオープン ソース ライセンスに適合している証拠を、リリースごとに反復して、ソース コードスキャンを実施します。

監査の目的は：

- 最後のスキャンからの間に、加えられた（削除された）オープン ソースを説明するリリース BOM を更新します。
- オープン ソースの由来も含めて、ソース コードの由来を確認します。
- 全ての依存関係、コード一致、ライセンス矛盾にフラグを付けます。

監査フェーズの前提条件

適切なコンプライアンス記録（チケットとも呼ばれます）は、特定のオープン ソース利用に関する全ての必要な情報を補足し、内部ビルドシステム内のソース コードの場所を提供するように、作成されます。ある場合には、特にプラットフォームのフル スキャンが実施される時、適切なコンプライアンス報告が作成される前に、オープン ソース コンポーネントはスキャンされることもあります。この場合には、オープン ソース コンポーネントが発見された時に、記録が作成されます。

監査フェーズの結果

- ソース コードの出自とライセンスを特定する監査報告
- 変更要求チケットは、適切な技術チームに対して、監査中に特定された解決すべき課題について記載されます。

いくつかの活動が、ソフトウェア コンポーネントの発見と監査の契機となります。（図 12、次ページ）

- 開発者からのオープン ソース コンポーネント利用の要求.
- 全ソフトウェア スタックのソース コード スキャン
- 以前に承認されたコンポーネントでのソース コード変更サード パーティ ベンダーから受け取ったオープン ソースウェブからダウンロードされたソース コード（著者やライセンスが不明）
- ソース コード リポジトリへ提出されるプロプライエタリ ソフトウェア
- コード リポジトリへ追加された、利用申請フォームに対応していないオープン ソース コード リポジトリへ追加された、利用申請フォームに対応していないオープン ソース
- 以前、異なる製品で承認を受けたオープン ソースの利用

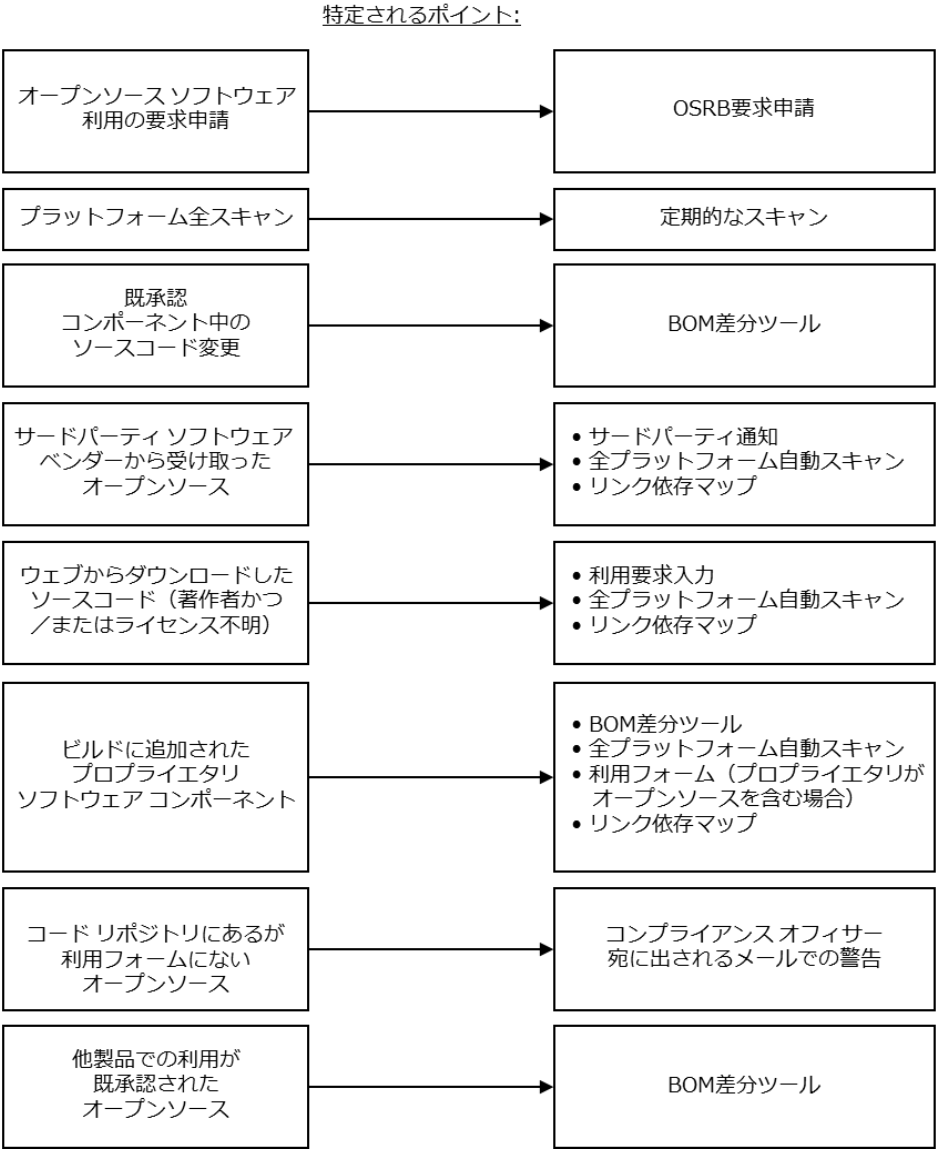


図 12：入るオープン ソースの特定と監査の方法

ステップ 3 — 課題解決

コンプライアンス精査のこのステップでは、監査ステップで特定された全ての課題が解決されます。OSRB チェアは、監査ステップで技術者に割り当てられたチケットのクローズを監視します。技術者が特定された課題を解決した時には、OSRB チェアは、解決された課題がもはや存在しないことを確認するために監査を要求するべきです。

課題解決フェーズの前提条件

ソース コード スキャンが完了し、ソース コードの出自とライセンスを特定した監査報告が生成される。報告は、特定されていないソース コード ファイルや、異なるライセンスで結合されるソース コードによって生じるライセンス矛盾にフラグを付けます。コンプライアンス オフィサーは、これらの課題解決を推進します。

課題解決フェーズの結果

報告にあるフラグ付きのファイルの解決と、フラグ付きのライセンス矛盾の解決

ステップ 4 — レビュー

監査が完了し、以前に特定された全ての課題が解決されたら、特定のソフトウェア コンポーネントに対するコンプライアンス チケットは、レビュー ステップへ移動します。図 13（次ページ）に示すように、さまざまなレビューが実施され、特定された全ての課題は解決していなければなりません。レビュアーは、ソフトウェアの利用、変更、頒布を支配するライセンスを理解し、さまざまなライセンスの義務を特定する必要があります。与えられたソフトウェア コンポーネントに対する、コンプライアンス チケットのレビュアーは、

- 社内パッケージ所有者（特定のソース コード コンポーネントの開発者）
- ソース コード スキャンか監査の担当者

- OSRB チェア、法務担当、および OSRB 技術代表からなる OSRB (オープン ソース レビュー ボード)
- OSEC （オープン ソース エグゼクティブ ボード）

図 13 コンプライアンス チケットのレビュアーとその役割

コンプライアンス精査のこのステップの一部として、2 つの重要なレビューがあります：アーキテクチャー レビュー、リンク解析レビュー

アーキテクチャー レビュー

アーキテクチャー レビューの目的は、オープン ソース、サード パーティ、プロプライエタリ コード間の相互作用を解析することです。アーキテクチャー レビューの結果は、オープン ソース コンポーネントからプロプライエタリ コンポーネントへ（その逆も）影響を与えるライセンス義務の解析です。社内パッケージ所有者、OSRB 技術代表、そしてオープン ソース エキスパートが、アーキテクチャー レビューを行います。もしライセンスの矛盾になるような依存関係を発見した場合、コンプライアンス オフィサーは、ソース コードの再作業によって依存関係問を題解決するためにチケットを技術担当に発行する。

リンク解析レビュー

リンク解析の目的は、動的および静的リンク レベルで、GPL ライブラリをプロプライエタリ ソース コード コンポーネントにリンクしているなど、問題発生の可能性のあるコードの組み合わせ を検出することです。OSRB チェアは、自動ツールを使用してこの調査を行います。リンク問題は、解決するように技術担当に報告されます。

レビュー フェーズの前提条件

監査されたソース コード、解決されている全ての課題。

レビュー フェーズの結果

OSRB メンバーは、特定のコンポーネントに対してアーキテクチャー レビューとリンク解析を実施し、もし課題が発見されなければ、次のステップ（つまり承認）への準備が整っているという印をつけます。

ステップ 5 — 承認

全てのレビューが完了したら、ソフトウェア コンポーネントのコンプライアンス チケットは、製品に利用して良いかを決定する承認ステップへ移動します。承認は OSRB が行います（前の章で説明したように、法務担当、技術代表、オープン ソース エキスパートが含まれます）。

ほとんどのソフトウェア コンポーネントに対して、チケットがコンプライアンス プロセスのこの点まで進めば、承認が与えられます。OSRB がオープン ソース コンポーネントの利用を承認したら、OSRB は製品チームに承認を伝えますので、製品チームは責任を理解し、ライセンス義務履行の準備を始めます。OSRB がオープン ソース コンポーネントの利用を却下する場合、OSRB は却下の理由を要求者に伝え、この情報はコンプライアンス チケットの一部として記録されます。要求者は OSRB に再考を促すことは可能ですが、結果としては、オープン ソース コンポーネントは製品に使用できません。

承認フェーズの前提条件

全ての OSRB メンバーはコンプライアンス チケットをレビューし、OSRB はアーキテクチャー レビューとリンク解析を完了している。

承認フェーズの結果

特定コンポーネントの利用の承認または却下

ステップ 6 — 登録

ソフトウェア コンポーネントが製品やサービスへの利用を承認されたら、コンプライアンス チケットは、承認を反映するように更新されます。ソフトウェア コンポーネントは、オープン ソース利用とユースケースを記録したソフトウェア一覧表に加えられます。

保守的なアプローチの後にコンプライアンス プラクティスを進めるならば、特定のバージョンと、特定の製品やサービスバージョンでの利用に対して、オープン ソース ソフトウェアを承認します。このオープン ソース ソフトウェアの新しいバージョンが入手可能になったら、利用形態とライセンスが内部ポリシーと整合していることを確認するために新しい承認依頼を出します。

登録フェーズの前提条件

OSRB はコンポーネントの製品への利用を承認している。

登録フェーズの結果

コンポーネントは、コンポーネント名、バージョン、内部所有者、および製品名、バージョン、リリース番号等のコンポーネントが利用される場所の詳細などと共に、ソフトウェア一覧表に登録されます。

ステップ 7 — 通知

オープン ソースを利用する際の重要な責務の一つは、ドキュメント記載の義務、または通知義務ともいわれるものです。外部に提供する製品やサービスにオープン ソースを利用している企業は以下を行う必要があります：

- エンド ユーザーに、ライセンス義務を満たした結果として入手可能になっているソース コードのコピーをどのおうに入手できるかを通知します（必要に応じて）
- 必要な著作権や帰属を通知することで、オープン ソースを利用していることを知らせます。
- 製品に含まれているオープン ソース コードに対応するライセンスの全文を提示します。

オープン ソース ライセンス義務を順守しない企業は、ライセンスを取得できず、著作権を侵害したとして、著作権所有者に法的行動をさらされる可能性があります。また、対象となるソフトウェアを使用および頒布する権利を失う可能性があります。ドキュメント記載義務を履行するには、製品に適切な通知を含める必要があります。コンプライアンス精査の中のこのステップでは、OSRB チェアが通知文を準備し、それを各担当部署に手渡します 。

通知フェーズの前提条件

ソフトウェア コンポーネントについて、使用が承認され、ソフトウェア一覧表への登録が完了している。

通知フェーズの結果

対象のコンポーネントのライセンス、著作権、および帰属通知が準備され、製品の資料に含めるべく担当部署に手渡される。

ステップ 8 — 頒布前検証

コンプライアンス精査の次のステップは、頒布の方法とモード、頒布するパッケージのタイプ、および頒布メカニズムを決定することです。

頒布前検証の目的は、以下を確実にすることです：

- 頒布されることになっているオープン ソース パッケージは、特定され、承認されている。
- ソース コード パッケージ（変更を含む）は、製品として出荷されるバイナリと一致することが確認されている。
- 適切な通知が、製品文書に含まれていて、エンド ユーザーに特定のオープン ソースのソース コードを要求する権利があることを知らせている。
- 全てのソース コード コメントはレビューされ、不適切な内容は取り除かれている。これは厳密にはコンプライアンスの課題ではないですが、しかしながら、ある場合には、コードが受け取られた場所に関する無邪気なコメントが大きなコンプライアンスに関する問い合わせを生むきっかけになる可能性があります。

頒布前検証フェーズの前提条件

コンポーネントは利用が承認され、ソフトウェア一覧表に登録され、全ての通知が収集され義務履行のために送付されている。

頒布前検証フェーズの結果

- 頒布方法と様式を決定します
- 全ての頒布前検証が問題なく完了していることを確実にします。

ステップ 9 — 頒布

頒布前検証が全て完了したら、対応する製品とバージョンをラベル付けして（このシナリオは、ソース コードを入手可能になることを想定しています。他の方法は後続の章で議論します。）、オープン ソース パッケージを頒布用ウェブサイトにアップロードします。この作業は、コードのダウンロードを希望する人には役に立ちますが、これ自体でライセンス義務を満たしていることにはなりません。さらに、推奨プラクティスでは、コンプライアンスやオープン ソースに関係する問い合わせの受付に関する、電子メールや住所などの情報を提供します。

頒布フェーズの前提条件

全ての頒布前検証がチェックされ、問題が発見されていない。

頒布フェーズの結果

対象のコンポーネントのソース コードは、頒布用のウェブサイトにアップロードされる。（そういう頒布方法が選択された場合）

ステップ 10 — 最終検証

オープン ソース パッケージを頒布用のウェブサイトにアップロードしたら、パッケージが正しくアップロードされ、ダウンロードでき、エラー無しに外部コンピューターで解凍できることを確認します。もし、パッチを提供する時には、それが容易に適用できること、アップストリーム コンポーネントの適切なバージョンを特定していることを確認します。

最終検証フェーズの前提条件

ソース コードはウェブサイト上で公開される。

最終検証の結果

ソース コードは正しくアップロードされ、ダウンロード可能で、承認されたものと同じバージョンに対応しているという確認結果を受けます。

第 5 章

コンプライアンス プロセスとポリシー

本書の目的から、ここでの議論の焦点は、商用製品におけるオープン ソースとプロプライエタリやサード パーティ ソース コードとの統合や利用に当てます。ここでの議論では、テストや評価目的など組織内だけで利用されるオープン ソースのためのポリシーやプロセスは除外されますこの章では、基本コンプライアンス プロセスとインクリメンタル コンプライアンス プロセスや、インクリメンタル コンプライアンスを達成するためのガイドラインに加えて、利用ポリシーや利用プロセスも議論します。

ポリシー

利用ポリシーは、コンプライアンス プログラムの不可欠要素です。このポリシーは、長文や複雑なものである必要はありません。単純なポリシーでも、以下の項目を要求するものであれば、複雑なポリシーと同じように効果的であることができます。

- 技術者は、いかなるオープン ソースであっても製品に統合する前には、OSRB から承認を得ることが必要です。
- サード パーティ企業から受け取るソフトウェアは、含まれているオープン ソースを全て特定しなければなりません。このことは、製品出荷前までにライセンス義務を実施することを確実にしてくれます。
- 全てのソフトウェアは、監査とレビューを受ける必要があります。これにはプロプライエタリ ソフトウェア、サード パーティ プロバイダーから受け取ったソフトウェア、そしてオープン ソース ソフトウェアを含みます。
- 製品は、顧客が受け取る前に、オープン ソース ライセンス義務を実施しておく必要があります。
- 一つの製品に対する承認は、たとえ同じオープン ソース コンポーネントを利用していても、別の製品への承認とはなりません。
- 全ての変更されたコンポーネントは、承認プロセスを通過しなければなりません。

これらの規則は、いずれのソフトウェア（プロプライエタリ、サード パーティ、オープン ソース）も製品基盤へ導入されるまでに監査され、レビューされ、承認されることを確実にします。さらに、それは、顧客が製品を受け取る前に、さまざまなソフトウェア コンポーネント利用に起因するライセンス義務を実行する計画を、企業が持つことを確実にします。

プロセス

コンプライアンス利用プロセスは、問題となるソフトウェア パッケージのソース コードのスキャン、発見された課題の特定と解決、法務レビューおよびアーキテクチャー レビューの実施、そのソフトウェア パッケージに対する利用許可に関する判断を含みます。

図 14 は、コンプライアンス利用プロセスの簡略化された図を示しています。この図は、プロセスの反復的な性質は示していません。より詳しい図は、図 17（90 ページ）に示します。

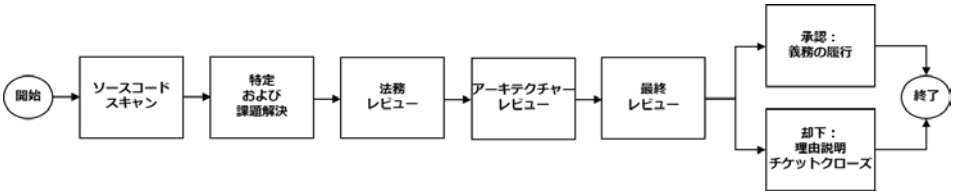


図 14. コンプライアンス利用プロセスの例

ソース コード スキャン

ソース コード スキャン フェーズで、全てのソース コードはソース コード スキャン ツールを使用してスキャンされます。図 15（82 ページ）は、ソース コード スキャンを開始する要因を示しています。それらは、

- 通常、技術スタッフによって記入される OSRB 利用フォームの到着。これは、技術者や開発者が、問題となるソース コードについての基本情報を提供するために記入する簡単なオンラインのフォームです。フォームの提出によって、（JIRA や Bugzilla などのシステムで）コンプライアンス チケットが自動的に生成され、ソース コード スキャン要求が監査スタッフへ送られます。
- 定期的に計画された全プラットフォーム スキャン：こうしたスキャンは、ソフトウェア プラットフォームに OSRB フォーム無しで忍び込んでいるかもしれないオープン ソースを明らかにするのに非常に役に立ちます。
- 前に承認されたソフトウェア コンポーネントに対する変更：多くの場合、技術スタッフは特定のバージョンの OSS コンポーネントを評価やテストして、新しいバージョンが利用可能になった時にそのコンポーネントを採用します。
- オープン ソースを開示しているかわからないサード パーティ ソフトウェア プロバイダーから受け取ったソース コード。
- 作者やライセンスが不明であり、オープン ソースが含まれているかもしれないウェブからダウンロードしたソース コード。
- 技術者がオープン ソース コードを模倣しているかもしれないし、プロプライエタリ ソフトウェアにオープン ソース コード使っているかもしれないようなビルドシステムに、入ってくる新しいプロプライエタリ ソフトウェア コンポーネント。

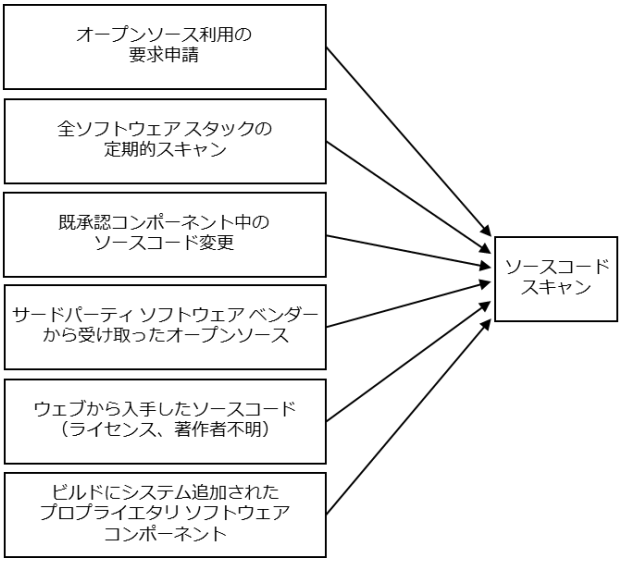


図 15：ソース コード スキャンを開始するイベント

法務レビューへの準備中に、ソース コード スキャンを実施する担当者は、著作権や帰属を表記しているファイルに加えて、COPYING、README、LICENSE ファイルなど、パッケージで見つかった全てのライセンス情報をコンプライアンス チケットに付けておくべきです。

特定と解決

特定と解決フェーズで、監査チームは、スキャン ツールでフラグをつけられたファイルやコード断片を検査し解決します。

法務レビュー

法務レビュー フェーズでは、法務担当（OSRB メンバー）は、スキャン ツールによって生成された報告や、ソフトウェア コンポーネントのライセンス情報や、技術者や OSRB メンバーによってコンプライアンス チケットに記載されたコメントをレビューします。ライセンスに問題が無いならば、法務担当は、ソフトウェア コンポーネントに関して、入るライセンスと出るライセンスを決定し、コンプライアンス アーキテク

チャー レビュー フェーズへコンプライアンス チケットを進めます。例えば両立しないライセンスをもつソース コードを組み合わせるなど、ライセンスに課題が発見されたときには、法務担当はこれらの課題にフラグをつけて、コードの再作業のためにコンプライアンス チケットを技術担当へ再割り当てします。ライセンス情報が不明確である、ライセンス情報が入手できない等、いくつかの場合には、法務担当は、不明瞭な点を明確にするとともに特定のソフトウェアに付与されているライセンスの確認をするために、プロジェクトのメインテナーやオープン ソース開発者にコンタクトします。

アーキテクチャー レビュー

アーキテクチャー レビューの中では、コンプライアンス オフィサーと OSRB 技術代表は、オープン ソースとプロプライエタリ、およびサード パーティ コードの間の相互関係を解析します。これは特定されたアーキテクチャー図を検査することで実行されます。

- オープン ソース コンポーネント (そのままで利用、又は変更して利用)
- プロプライエタリ コンポーネント
- サード パーティ ソフトウェア プロバイダー由来のコンポーネント
- コンポーネントの依存状態
- 通信プロトコル
- 特に、異なるオープン ソース ライセンスによって許諾されているような場合で、特定のソフトウェア コンポーネントが相互作用するか、または依存するような、他のオープン ソース パッケージ。

アーキテクチャー レビューの結果は、オープン ソースからプロプライエタリやサード パーティ ソフトウェア コンポーネントへ（同様にオープン ソース コンポーネント間で）影響を与えるようなライセンス義務の分析となります。コンプライアンス オフィサーは、例えば、プロプライエタリ ソフトウェア コンポーネントと GPL で許諾されたコンポーネントがリンクされるような課題を見つけたら、解決のためにコンプライアンス チケットを技術担当へ送ります。課題が無いならば、コンプライアンス オフィサーは、チケットを承認プロセスの最終段階へ送ります。

最終レビュー

最終レビューは、通常は OSRB メンバーが対面する会議で行い、OSRB は利用の許可または却下を決めます。ほとんどのケースでは、ソフトウェア コンポーネントが最終レビューの段階まで到達したら、（そのソフトウェア コンポーネントがもう使われていないなどの）何かの条件が現れない限り、そのコンポーネントは承認されます。承認されたら、コンプライアンス オフィサーは、承認されたソフトウェア コンポーネントのライセンス義務リストを準備し、義務実施のために適切な部署へ送ります。

プロセスの各ステージにおけるインプットとアウトプット

この節では、図 16 で示される OSRB 利用プロセスの 5 つのフェーズにおけるインプットとアウトプットについて議論します。これらのフェーズは、説明目的のものであって、実際のシナリオとは正確には一致しないかもしれません。

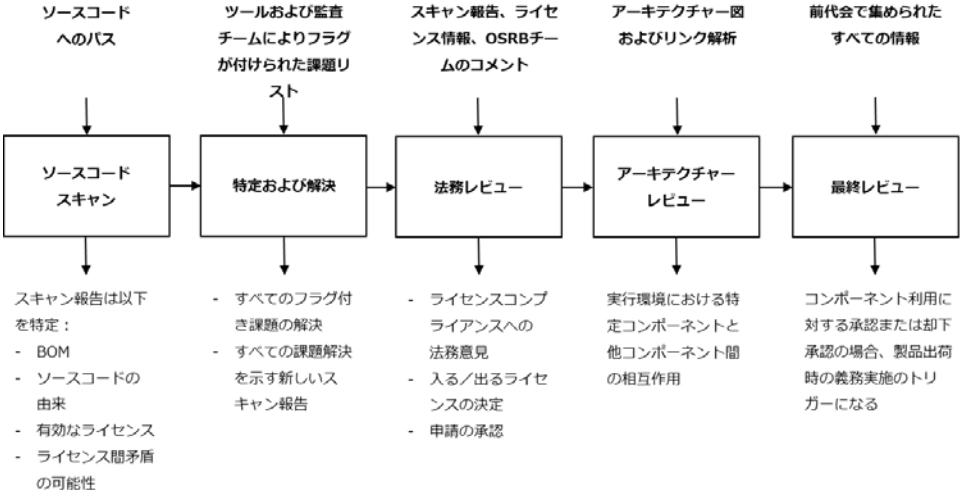


図 16 利用プロセスのインプットとアウトプット

ソース コード スキャン段階

インプット

スキャン フェーズへのインプットは、技術者がオンラインで記入して提出する OSRB 利用フォームとなります。表 6 (98 ページ)はフォームの詳細です。フォームは、ソース コード リポジトリ システムの中でのソース コードの位置も含めて、問題となっているオープン ソース コンポーネントに関する全ての情報を含んでいます。定期的な全プラットフォーム スキャンは、対応する OSRB フォーム無しにプラットフォームへソフトウェア コンポーネントが入らないことを確実にするために、数週間に 1 回程度実施されるべきです。

アウトプット

スキャン フェーズのアウトプットは、ソース コード スキャン ツールによって生成される報告です。これには、次のような情報が書かれます。

- 利用されている既知のソフトウェア コンポーネント、ソフトウェア構成リスト（BoM）として知られているもの影響があるライセンス、ライセンス文、義務の要約法務担当によって検証されるべきライセンス矛盾
- ファイル一覧表
- 特定されたファイル
- 依存関係
- コード一致
- 特定保留中のファイル特定保留中のソース コード一致

特定と解決フェーズ

インプット

このフェーズへのインプットは、前フェーズでスキャン ツールによって生成された報告書になります。この報告書は、矛盾を起こしているライセンスや両立しないライセンスのような課題にフラグを付けています。課題がないならば、コンプライアンス オフィサーは、コンプライアンス チケットを法務レビュー フェーズへと進めます。. 解決すべき課題がある時には、コンプライアンス オフィサーは、コンプライアンス チケットにサブタスクを設定し、解決のために適切な技術者に担当を割り当てます。ある場合にはコードの再作業が必要となりますし、別の場合には単に状況を明確にするだけで解決することもあります。サブタスクは、課題の説明、技術担当によって実施されるべき解決提案、そして完了までの具体的な日程を含んでいるべきです。

アウトプット

このフェーズでは、サブタスクはクローズされた上で、全課題の解決がアウトプットとなります。コンプライアンス オフィサーは、ソース コードの再スキャンを依頼してスキャン報告書を作成し、前に上がった課題が解決済みになっていることを確認するかもしれません。コンプライアンス オフィサーは、レビューと承認を受けるために、コンプライアンス チケットを法務代表へ送ります。

法務レビュー

インプット

コンプライアンス チケットが法務レビュー フェーズへ届いた時には、コンプライアンス チケットは以下を含んでいます。

- ソース コード スキャン報告、スキャン フェーズで発見された課題が全て解決済みであることの確認
- チケットに付けられたライセンス情報のコピー：通常、コンプライアンス オフィサーは、ソース コード パッケージに含まれている README、COPYING、AUTHORS などのファイルをコンプライアンス チケットに付

与します。通常 COPYING、LICENSE ファイルに含まれている OSS コンポーネントに関するライセンス情報以外にも、著作権や帰属情報を入手することが必要です。この情報は、製品ドキュメントに適切に入れられます。 コンプライアンス チケットに関するコンプライアンス オフィサーからの（懸念点や追加の質問などの）フィードバック

- OSRB の技術代表や、内部的にこのパッケージを担当／保守する技術者（パッケージのオーナー）からのフィードバック

アウトプット

このフェーズのアウトプットは、コンプライアンスに関する法務の意見、問題となっているソフトウェア コンポーネントについての入るライセンスと出るライセンスに対する決定となります。ソフトウェア コンポーネントは、異なるライセンスによって許諾されるソース コードを含むことがあるので、入るライセンスと出るライセンスは、複数のフォーム入ることがあります。

入るライセンス、出るライセンス

入るライセンスは、受け取ったソフトウェア パッケージに対するライセンスです。出るライセンスは、使用許諾するソフトウェア パッケージに対するライセンスです。あるケースでは、入るライセンスが（BSD のように）再許諾を許すような許容型ライセンスである場合には、企業はそのソフトウェアをプロプライエタリ ライセンスで再許諾します。より複雑な例としては、プロプライエタリ ソース コード、ライセンス A で許諾されているがライセンス B でも許諾可能なソース コード、ライセンス C で許諾されたソース コードが含まれるものがあります。法務レビュー期間に、法務担当は出るライセンスと入るライセンスを決める必要があります。

入るライセンス = プロプライエタリ ライセンス + ライセンス A + ライセンス B + ライセンス C

出るライセンス = ?

アーキテクチャー レビュー

アーキテクチャー レビューの目的は、オープン ソース コードとサード パーティ、プロプライエタリ コードの間の相互作用を解析することです。アーキテクチャー レビューの結果は、オープン ソース コンポーネントがプロプライエタリ コンポーネントへ影響を与えるかもしれないライセンス義務の分析です。企業内のパッケージ オーナー、OSRB 技術代表とコンプライアンス オフィサーが、通常アーキテクチャー レビューを実施します。もしライセンスの矛盾になるような依存関係を発見した場合、コンプライアンス オフィサーは、ソース コードの再作業によって依存関係問を題解決するためにチケットを技術担当に発行する。

インプット

監査されたソース コード、解決されている全ての課題。

アウトプット

OSRB メンバーは、特定のコンポーネントに対してアーキテクチャー レビューを実施し、次のステップ（つまり最終承認）への準備が整っているという印をつけます。

最終承認フェーズ

インプット

このフェーズへのインプットは、ソフトウェア コンポーネントに関する全てのコンプライアンス記録です。それには以下が含まれます。

- スキャン ツールによって生成されたソース コード スキャン報告
- 発見された課題、課題が解決された経緯に関する情報、課題が解決されたことを確認した人のリスト
- アーキテクチャー図、ソフトウェア コンポーネントが他のソフトウェア コンポーネントとどのように相互作用するかの情報

- コンプライアンスに関する法務の意見、入るライセンスと出るライセンスに関する決定
- 組み込み環境（C/C++）で適用可能であれば、動的リンクと静的リンクの解析

アウトプット

このフェーズのアウトプットは、ソフトウェア コンポーネントの利用に関する、承認または却下の判断です。

詳細な利用プロセス

勿論、コンプライアンス手順に影響を与える多くの事情があります。図 17（次ページ）は、いくつかの可能なシナリオと、コンプライアンス プロセスにおいて、一つのステップから別のステップへどのように状態遷移するかについて、詳細なプロセスを示しています。8 つの可能なプロセスを議論します。これらのシナリオは、相互に排他的ではありませんし、唯一の可能なプロセスでもありません。これらは、図示と議論の目的のために提示します。

シナリオ 1：スキャンされたソース コードは 100%プロプライエタリである

シナリオ 2：スキャンされたソース コードは両立しないライセンスを含むコードを含んでいる

シナリオ 3：リンクに関する課題がアーキテクチャー レビューの最中に特定される

シナリオ 4：ソース コード パッケージはもはや使われていない

シナリオ 5：精査でライセンス義務を満足する形でリリースするべき IP が特定される

シナリオ 6：検証ステップで解決すべき課題が特定される

シナリオ 7：ソース コードは利用が承認される

シナリオ 8：ソース コードは利用が却下される

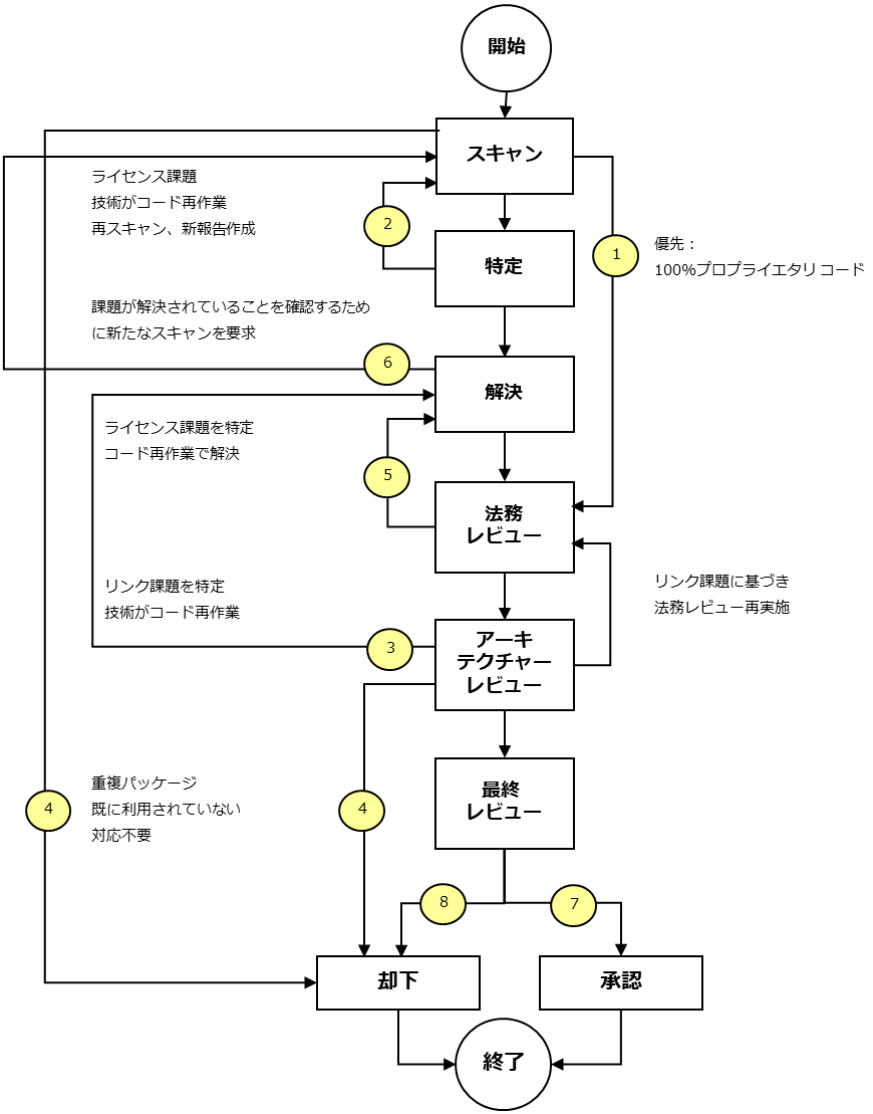


図 17：コンプライアンス シナリオの一例

シナリオ 1：スキャンされたソース コードは、100%プロプライエタリである

スキャンされたソフトウェア コンポーネントは、100%プロプライエタリ コードだけを含んでいて、オープン ソース コードは申告も特定もされていない。このケースでは、迅速な処理を想定し、特定のコンポーネントのコンプライアンス チケットは、法務レビューに進められます。法務担当は、ライセンスをこのプロプライエタリ コンポーネントに付与し、アーキテクチャー解析とリンク解析の実施のためにコンプライアンス オフィサーへ送ります。

シナリオ 2：両立性のないライセンス（の組み合わせ）

スキャンされたソフトウェア コンポーネントは、両立性のないライセンスが付けられた複数のソース コードに由来するソース コードを含んでいることがあります。他の例としては、プロプライエタリ ソース コードと GPL で許諾されたソース コードとが組み合わされたものがあります。このシナリオでは、コンプライアンス チケットにスキャン報告が付けられ、プロプライエタリ ソフトウェア コンポーネントから GPL ソース コードを取り去るという作業要求と共に、企業内でそのソフトウェア コンポーネントを管理している開発者に作業が割り当てられます。いったん開発者がコードの再作業を行った後は、そのソフトウェア コンポーネントは、法務レビューへ進む前に、GPL コードが取り除かれたことを確認するために、再スキャンされます。

シナリオ 3：リンクに関する課題が特定される

このシナリオでは、コンプライアンス チケットは、法務レビューをパスして、アーキテクチャー レビューとリンク レビューに進みます。コンプライアンス オフィサーは、リンクに関する課題を見つけます。このケースでは、コンプライアンス オフィサーは、コンプライアンス チケットを解決フェーズに送り、リンクに関する課題を解決するように開発者に割り当てます。

シナリオ 4：ソース コードは、もはや使われない

このシナリオでは、技術担当は、コンプライアンス プロセスの処理中に、そのソフトウェア コンポーネントが製品に混入しないようにします。結果として、そのコンプラ

イアンス チケットは（却下として）クローズされます。当該コンポーネントが次に利用される場合には、そのコンポーネントが製品に組み込まれるか、ソース コード リポジトリに入る前に、コンプライアンス プロセスに再入力して承認を得なければなりません。

シナリオ 5：開示要求のリスクがある IP

このシナリオは、法務レビューが、非開示になっている知的財産とオープン ソース コード パッケージとが組み合わされていることを明らかにしたものです。法務担当は、この件にフラグを付けて、オープン ソース コンポーネントからプロプライエタリ ソース コードを取り除くために、コンプライアンス チケットを技術担当に割り当てます。技術担当がオープン ソース コンポーネント中にプロプライエタリ ソース コードを保持し続けることを固辞した場合には、OSEC はプロプライエタリ ソース コードをオープン ソース ライセンスの下でリリースする決定をしなければならなくなるでしょう。

シナリオ 6：未解決の課題が見つかる

OSRB メンバーがソフトウェア コンポーネント中にコンプライアンスに関する課題を発見したような全てのケースは、コンポーネントは同じライフ サイクルを進みます。

- 技術担当は、特定された課題を解決します。
- 監査チームは、ソフトウェア コンポーネントを再スキャンし、新しいスキャン報告を提出します。
- 法務担当は、新しい監査報告を確認します。
- コンプライアンス オフィサーは、アーキテクチャー解析、リンク解析において未解決の課題がないことを確認します。

シナリオ 7：ソース コードは、承認される

ソフトウェア コンポーネントが、監査、法務レビュー、コンプライアンス承認を全て受けたら、OSRB 会議でその結果をレビューします。もし、状況に何も変更が無いならば、つまり、まだ使用されており、同じバージョンであり、利用形態が同じであること（次ページの図 18）コンプライアンス オフィサーは、

- ある OSS ソフトウェア コンポーネントのバージョン X は、製品 Y のバージョン Z での使用が承認されたことをソフトウェア一覧表に反映させます。
- 製品に付属する文書中のエンド ユーザーへの通知を更新して、製品やサービスにオープン ソースが使用されていることを反映させるため、文書チームへチケットが発行されます。
- 製品出荷前に、頒布プロセスを開始させます。

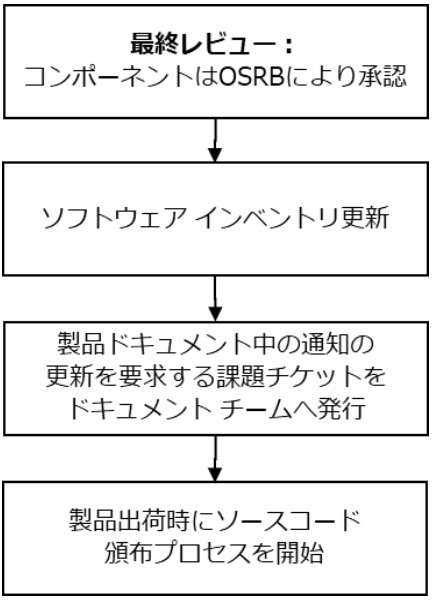


図 18：OSRB 承認後に実施されるステップ

シナリオ 8：ソース コードは利用が却下される

このシナリオでは、OSRB は、特定のソフトウェア利用の却下を決定します。そういう却下につながるような理由がいくつかあります。

- そのソフトウェア コンポーネントは、もはや使われない。
- 容易に解決できないリンクに関する課題がある。このケースの結論は、開発を中止し、より良いソリューションを設計しなおすというものです。

- 容易に解決できない両立しないライセンスがある。このケースの結論は、開発を中止し、より良いソリューションを設計し直すというものです。
- 特定のコンポーネントの利用やリリースを阻害する知的財産に関する課題がある。
- その他の理由：それぞれのケースは、問題となっているソフトウェア コンポーネントの個別条件や、最終製品やサービスでの利用形態に依存します。

インクリメンタル コンプライアンス プロセス

インクリメンタル コンプライアンスは、すでに初期コンプライアンスが完了しているベースライン版に製品フィーチャーを追加する時に、コンプライアンスを維持するために実施されるプロセスです。（ベースライン コンプライアンスとも呼ばれる、初期コンプライアンスは、開発が開始された時に発生し、製品の初期版が出荷されるまで継続します。）インクリメンタル コンプライアンスは、ベースライン コンプライアンスを確立するために必要とする努力と比べると小さな努力ですみます。

図 19 は、製品開発とインクリメンタル コンプライアンスを示しています。

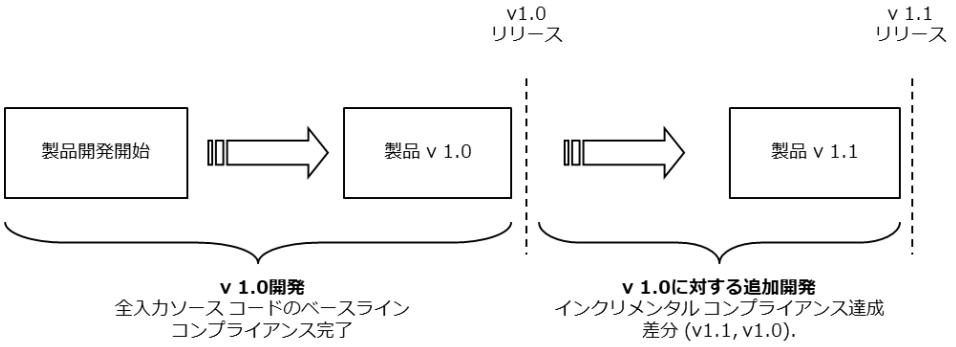


図 19：インクリメンタル コンプライアンス

この例では、コンプライアンス チームは、ソフトウェア ベースライン（ここでは仮に V 1.0 と呼びます）に含まれる全てのオープン ソースを特定し、オープン ソース コンポーネント全てが、コンプライアンス プロセス全体を通るようにします。製品が出荷されたら、追加機能や不具合修正などを含む開発は、新しいブランチで行われます。この例では、V 1.1 です。

インクリメンタル コンプライアンスに関しては、解決すべきいくつかの課題が発生します。特に、V1.0 と V1.1 の間で変更されたソース コードを正確に把握し、リリースの間の変化点についてコンプライアンスを検証します。

- 新しいソフトウェア コンポーネントが、導入されているかもしれません。
- 既存のソフトウェア コンポーネントが、既に使用中止になっているかもしれません。
- 既存のソフトウェア コンポーネントが、新しいバージョンに更新されているかもしれません。
- ソフトウェア コンポーネントのライセンスが、バージョンの間で変更されているかもしれません。
- 既存のソフトウェア コンポーネントが、不具合修正を含むコード変更や、機能やアーキテクチャーに対する変更を含んでいるかもしれません。

あるべき質問は、「どうすれば全ての変更を追跡できるだろうか？」というものです。

答えは単純です。：第 7 章で議論しますが、BoM 比較ツールです。ここでの議論のために簡単に言うと、そのツールは、同じ製品やサービスに使う 2 つの BOM の差分を与えてくれます。製品 V1.1 の BOM と製品 V1.0 の BOM があると、その差分を計算します。ツールの出力は以下のようなものです。

- V1.1 で追加された新しいソフトウェア コンポーネントの名前
- 更新されているソフトウェア コンポーネントの名前
- 使用されなくなったソフトウェア コンポーネントの名前

この情報を知ることで、インクリメンタル コンプライアンスの達成は、比較的容易なタスクになります。

- 新しいソフトウェア コンポーネントをコンプライアンス プロセスに入力します。
- 変更されたソフトウェア コンポーネントのソース コードの差異を計測し、再度ソース コードをスキャンするか、あるいは前のスキャンは信頼に足るものかを、決定します。
- もはや使われなくなったソフトウェア コンポーネントをソフトウェア管理表から除いて更新します。

図 20（次ページ）は、インクリメンタル コンプライアンス プロセスの全体像を示しています。それぞれの製品リリースに対する BOM ファイルは、ビルド サーバーに保存されます。BOM 差分ツールは、異なる製品リリースに対応する 2 つの BOM ファイルを入力として受け取り、前に議論したように、変更リストを生成します。この時点で、コンプライアンス オフィサーは、リリースに含まれる全ての新しいソフトウェア コンポーネントに対して新しいチケットを発行し、ソース コードが変更されている部分はコンプライアンス チケットを更新した上で、できればプロセスを再実行し、使用中止になったソフトウェア コンポーネントを承認リストから削除して、ソフトウェア管理表を更新します。

OSRB 利用フォーム

OSRB 利用フォーム（要求フォームとも呼ばれます）を記入することは、オープン ソース ソフトウェアを企業へ導入する際の最も重要なステップの一つです。ですから、真剣に取り扱うべきものです。開発者は、該当するオープン ソース コンポーネントの利用の承認を要求するオンライン フォームを記入します。そのフォームは、提案されているオープン ソース コンポーネントの利用を OSRB が承認または却下することを決めるのに必要な情報を与えるような、いくつかの質問から構成されています。図 6（97 ページ）は、OSRB 利用フォームで要求されている情報を示しています。通常、これらの値は、データ入力を効率的に行うために、プルダウン メニューで選べるようにします。

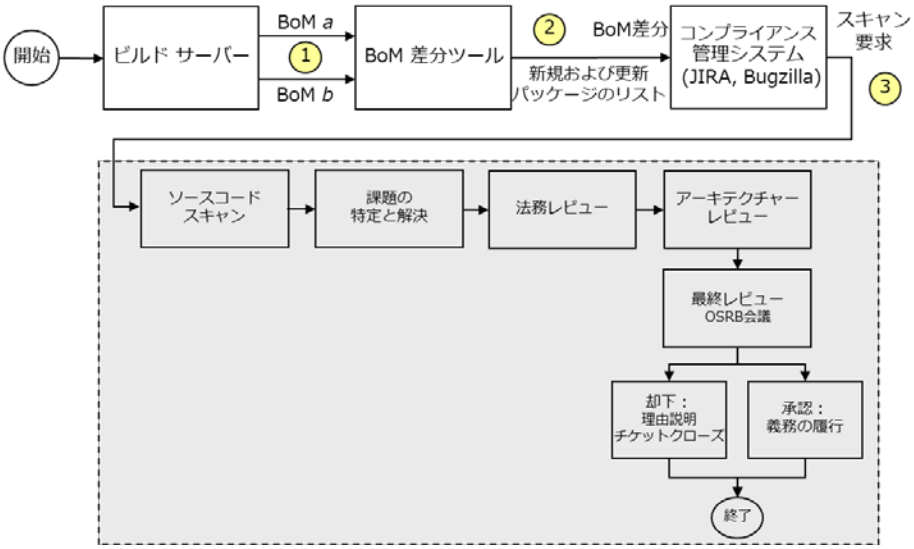


図 20：インクリメンタル コンプライアンス プロセスの例

ダウンロードされたオープン ソース パッケージに対する注意

ウェブからダウンロードしたオープン ソース パッケージをオリジナル形式で保管しておくことは不可欠なことです。これらのパッケージは、後の段階で（頒布の前に）、オリジナル パッケージと変更されたパッケージの差異を計算することによって、ソース コードに加えられた全ての変更を検証し追跡するために、使用されます。もし、サード パーティ ソフトウェア プロバイダーがオープン ソース コードを使っていれば、そのコードを製品に統合する製品チームは、OSRB 利用フォームに、利用するオープン ソースを記載して提出しなければなりません。もし、サード パーティ ソフトウェア プロバイダーが、ソース コード無しにバイナリだけを提供している場合には、製品チームと、そのサード パーティ ソフトウェア プロバイダーと連絡を取っているソフトウェア供給マネージャは、両方、またはどちらかが、（例えばスキャン報告などによって）提供されたソフトウェアにオープン ソースが含まれていないことを確認しなければなりません。

表 6：OSRB 利用フォームの一部として要求される情報

区分	説明
提出者情報	フォームを提出する社員の社員 ID（社員名、電話、E メール、管理者、職務場所、チームなどを社員管理表から受け取ります）
OSS コードの情報	OSS コードの情報 パッケージ名とバージョン ソフトウェア区分：オープン ソース、内製、サード パーティ パッケージ ウェブサイト URL 説明 ライセンス名とバージョン ライセンス ウェブサイト URL ソフトウェア カテゴリ：OS/kernel、driver、middleware、library、utility、other (explain)、etc. OSS コンポーネントを利用する利点 コンポーネントやパッケージとは別の手段 ソフトウェアを利用しないことによって発生すること SCMS でのソフトウェアの位置
ユースケース	内部利用（ツール、IT 等） 製品の一部として出荷 外部向けサービスを可能にする
変更	変更されているか (Y/N)? 会社 IP は含まれるか? IP を開示するか?

アーキテクチャー図に対する注意

アーキテクチャー図は、あるプラットフォーム上でのさまざまなソフトウェア コンポーネント間の相互作用を示します。図 21 は、アーキテクチャー図の一例です。この図では、以下を示します。

- モジュールの依存関係
- プロプライエタリ コンポーネント
- オープン ソース コンポーネント（変更後とオリジナル）
- 動的リンクと静的リンク
- カーネル空間とユーザー空間
- 共有されているヘッダ ファイル
- 通信プロトコル課題となっているソフトウェア コンポーネントが、相互作用する、または依存関係にあるような、他のオープン ソース コンポーネント。特に、別のオープン ソース ライセンスによって許諾されている場合。

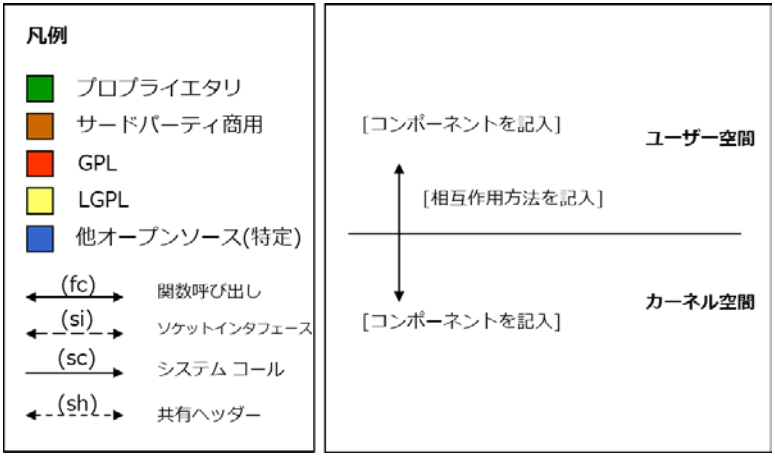


図 21：（C や C++に依存するような組み込み環境へ適用する）アーキテクチャー図のテンプレート

OSRB 利用フォームに影響する規則

OSRB 利用フォームに影響するいくつかの規則があります。ここにいくつか挙げます。

- フォームは、特定の利用状況における、特定の製品でのオープン ソースの利用に対して適用します。それは、全製品の全ユースケースに対するそのオープン ソース コンポーネントの全般的な承認ではありません。
- フォームは、監査活動の基礎となり、OSRB が、フォームで表現された利用計画と、監査やアーキテクチャー レビューの結果とに矛盾が無いかを確認するための情報を提供します。
- フォームは、該当するオープン ソース コンポーネントの利用計画が変更になった場合には、更新して再提出されなければなりません。
- 技術担当がそのオープン ソースを製品ビルドに統合する前に、OSRB がフォームを承認しなければなりません。
- ライセンスが特許許諾条項や非係争条項を要求している時には、OSEC がオープン ソース利用の承認をしなければなりません。

監査

良い監査プラクティスは、製品やサービスの一部として採用される全てのソフトウェアの由来を完全に把握することを確実にします。このことを理解することによって、オープン ソース ソフトウェアライセンス義務を果たす組織能力が出来上がります。監査ポリシーは、単純でわかりやすいものです：ポートフォリオ／スタックに含まれる全てのソース コードは、監査され、コンプライアンス チケットに監査報告がつけられなければなりません。監査プロセスは、技術担当が特定のソフトウェア コンポーネントに対する OSRB 利用フォームを提出した後に、続いて実行されるワークフローから構成されます。

監査プロセスは、次のフェーズで構成されます。（図 22）

- 監査すべきソース コードの場所を含む、OSRB 利用フォームを受け取ります。
- ソース コードのスキャンを実施します。. スキャン ツールでフラグがつけられたコンポーネントの分析を実施します。
- 最終監査報告を作成します。

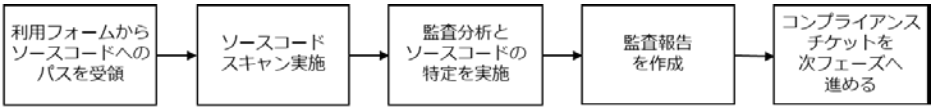


図 22：基本監査プロセス

ソース コード頒布

ソース コード頒布プロセスとポリシーの目的は、以下を確実にすることです。

- オープン ソース ソフトウェアを含む製品を購入する顧客やサービスを利用するユーザーは、適用可能な場合に、ソース コードを受け取る権利があることを通知される。
- 頒布したソース コードは、頒布サイトに転送されるソフトウェアのバイナリ バージョンと照らし合わせて正しいバージョンであり、また、適切にラベル付けされている。

頒布の動機

オープン ソース コードを頒布することには主に 3 つのビジネス上の動機があります。ライセンス義務の遵守、オープン ソース プロジェクトの価値を高めるための貢献、新しいオープン ソース プロジェクトに対するコードの貢献です。

ライセンス義務の順守

この例では、組織は、オープン ソースを製品やサービスに組み込んでおり、オープン ソース コンポーネント ライセンスにより、ソース コードに加えた変更を含めて、ソース コードの開示義務を持っています。コミュニティとの相互関係が双方向であるのに対して、このことは一方向頒布と略式に考えられます。

既存のオープン ソース プロジェクトへの修正の貢献

あるケースでは、オープン ソース ライセンスは、ライセンス コンプライアンスの目的からすると、修正を開示する義務を含んでいません。しかしながら、技術的な負荷、言い換えればこれらの修正を保守するコストを抑えるために、修正をリリースし、可能なアップストリームに載せます。

新しいオープン ソース プロジェクトの設立

組織は、ビジネス的な必要性から、新しいオープン ソース プロジェクトを設立し、ソース コードの貢献をするかもしれません。このケースは、既存のオープン ソース プロジェクトへ、（不具合修正や新規機能追加の形で）ソース コードの貢献をすることとは異なります。

頒布プロセスとポリシー

頒布ポリシーの目的は、ソース プロジェクト コード（＊＊＊）の供給プロセスに強い影響を与え、オープン ソース コードの入手性に関するオープン ソース ラインセンス義務の遵守するために、さまざまな流通[ロジスティックス]に対するガイドラインを提供することです。この頒布ポリシーは、ライセンスがソース コードの再頒布を要求するようなソフトウェア パッケージに適用されます。公開プロセス、公開方法、モード、チェックリストなどをカバーします。

プロセスを開始させる前に、ソース提供の方法と様式を決めなければなりません。プロセスは、外部へ頒布するソース コードの準備をもって始まり、頒布前チェックリストの確認、ソース コード パッケージの入手性の確認、そして、頒布後チェックリストの確認と続きます。

図 23 は、頒布プロセスの例を示します。以下を含みます。

- ソース コード提供方法を決めます。頒布様式を決めます。
- 外部頒布のためにソース コード パッケージの準備をします。
- 前に置かれた全ステップが問題なく完了していることと、外部頒布に向けてソース コード パッケージの準備が整っていることを確実にするために、頒布前チェックリストを全て確認します。
- 頒布を実行します。頒布プロセスの一部で発生するかもしれない誤りを把握するために、頒布後のチェックリストを確認します。

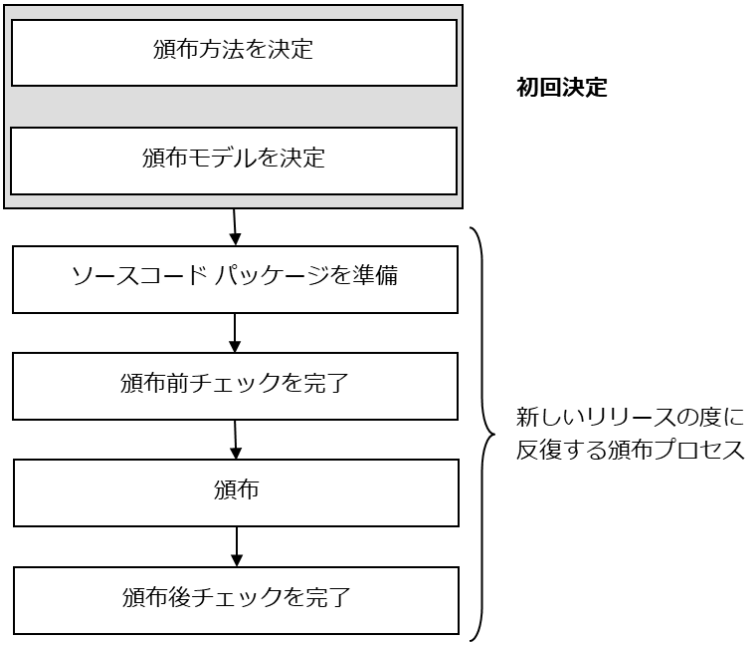


図 23：公開プロセスの例

頒布方法と様式

ソース コード パッケージを入手可能な状態にする、3 つの主な頒布方法があります。

即時コンプライアンス方法

この頒布方法に従い、製品やソフトウェア スタックの出荷時または直後にコードを提供し、一般的にはウェブサイトからダウンロードによって、コードを欲しい人は誰でも入手できるようにします。開発者は、入手資格を確認されることなく（つまりソース コードの入手資格を得るために製品を購入する必要なく）、直接ソース コードにアクセスが許されますので、このやり方は通常開発者にとって頒布の好ましい方法となります。あるケースでは、即時コンプライアンス方法は、メディアのディレクトリ product/device にソース コードを含めることで、達成することができます。

この頒布方法には 2 つの不利な点があります。まず第 1 に、全てのソース コードをパッケージ化し、製品出荷の準備を整える出荷日までに、ウェブサイト上でソース コードを入手可能にするという多大な努力が必要です。第 2 に、将来のソース コード頒布でも同じ方法が取られるだろうという期待を築いてしまいます。これは、毎回満足させるべきとても高い期待となります。

オンライン供給方法

この頒布方法によれば、ソース コードを入手する資格を持っている顧客だけが、排他的にアクセス権を与えられます。この方法は、アクセスしてソース コード パッケージをダウンロードするための認証を必要とするセキュアなウェブサイトにより最もよく管理されます。

オン デマンド コンプライアンス

この頒布方法は、オンライン供給方法の変形で、書面による通知(GPL/LGPL 系のライセンスの場合)を利用して、どのようにすればソース コードを要求したりアクセスしたりできるかを顧客に伝えます。ある組織は、企業窓口 email や郵便アドレス(書面による通知に書かれている)に対して書面で要求が送られるのを好みます。結果として、顧客は、入手資格を確認された後で、ソース コードのコピーを受け取ります。このコンプライアンス方法は、製品出荷後に、ソース コード パッケージ化終了のための追加の

予備時間を組織に与えてくれます。しかしながら、一般的に言って、ソース コードへのアクセスを要求してくる人たちの入手資格を確認するのにかかるオーバーヘッドや、要求を満たすために必要となる資源を考えると、好ましい頒布方法とは言えません。加えて、GPL/LGPL 系のライセンスに特有のケースとして、書面による通知は 3 年間有効でなければなりません。それゆえ、製品を最後に出荷した日から少なくとも 3 年間は、コード頒布を管理しなければなりません。CD-ROM でのソース コード頒布を選んだ場合には、追加のコストがかかりますし、ソース コード パッケージを格納した CD-ROM を生成するプロセスを確認する追加の検証ステップも必要になります。

頒布チェックリスト

顧客や一般向けのウェブサイトで公開する前に、オープン ソース パッケージの正当性を確認するための多くのチェックポイントがあります。さらに、追加の正当性確認は、一般に入手可能になった後にも必要になります。以下では、頒布前、頒布後のプロセスについて概要を示します。

頒布の前提条件

以下は、ソース コード パッケージが頒布可能になる前に、適合しておくべき条件リストです。（頒布の衛生とも呼ばれます）

- オープン ソース パッケージは、利用フォームで宣言された範囲で、その利用を OSRB によって承認されています。
- オープン ソース パッケージを含む製品は、出荷可能か、すでに出荷されています。
- もし、GPL/LGPL で許諾されたコードを開示するのであれば、自分たちが加えた変更について、コードと文書を提供することを確実にします。
- リンギスティック レビューを実施しています。このことはコンプライアンス関連ではないですが、将来使用予定の製品コード名、下品または粗野な言葉づかい、個人や email アドレスや URL への参照などがコードに残されたままであったというような課題が過去にありました。

頒布前チェックリスト

以下は、ソース コードの一般公開や頒布の前に行うチェックリスト例です。

- オープン ソース パッケージに加えられた変更は、文書化されて、オープン ソースのリリース ノートに変更履歴として含まれていることを検証します。変更されたソース コード ファイルは、著作権表記や免責事項の記載や一般的な変更履歴への記載を含んでいることを確認します。
- ソース コード パッケージに含まれている全ての内容物は、技術担当によってレビューされ、OSRB によって確認されていることを、確認します。
- オープン ソース パッケージは、企業のものでないマシンでコンパイルできることを確認します。企業で初期設定したマシン上でパッケージをコンパイルする時に、環境やコンパイラが全て事前に構成されたり設定されているということがよくあります。しかしながら、別のシステムでパッケージをコンパイルしようとする場合には、コンパイラの設定や、Makefile のオプションや、Include path 等が適切でないかもしれません。このステップの目的は、頒布しようとしているオープン ソース パッケージが、平凡なエンド ユーザー システム上でコンパイルできることを確実にすることです。
- 製品マニュアルを更新します。
  - 製品がオープン ソース ソフトウェアを含むことを言及します。
  - 製品に含まれる異なるオープン ソース ソフトウェアに対応する全てのライセンスのリストを含めます。
  - 適切な著作権と帰属に関する通知を提供します。
  - オープン ソース パッケージのコードへのアクセス方法（書面による申し出）、ウェブページからのダウンロードや、製品マニュアルで提供される特定のアドレス宛ての email や郵便による問い合わせなどを提示します。

- 書面による申出が、そういう通知を必要とするソース コード（基本的に、GPL/LGPL 系ライセンスで許諾されたコード）全てを包含しているかを検証します。
- ソース コードに不適切なコメントが残っていないことを確実にするために、リンギスティック レビューを実施します。ある企業では、リンギスティック レビューを通すことを忘れてしまい、製品がハックされた時に、ソース コード中に残っていた不適切なコメントが発覚し、困惑しました。リンギスティック レビューを実施する他の重要な理由は、ソース コードやコメントが、将来使用する開発コード名や機能を言及していないことを確実にすることです。
- 既存のライセンス表記、著作権表記、帰属表記に手が入っていないことを確実にします。
- 製品に搭載されて出荷されるバイナリに対応したソース コードであること、そのソース コードが製品とともに出荷されるライブラリを構成していること、ビルド手順がソース頒布（派生するバイナリは、タイム スタンプを除いて通常同一のものになります）に含まれていること、を検証します。
- パッケージが、OSRB 利用フォームで定義されたリンク関係や相互作用を守っていることを検証します。例えば、開発者がコンポーネントを LGPL で許諾されたライブラリと動的にリンクすると宣言したならば、その通りになっていて、代わりに静的リンクを使っていないことを検証する必要があります。これは、リンク依存関係マッピング ツールを使って検証されます。
- もし、まだ入っていなければ、オープン ソース パッケージのソース コード ルート フォルダーにある LICENSE ファイルに、ライセンス文のコピーを追加します。
- もし、ソース コード パッケージが、特別なビルド ツールや環境を必要とするならば、README ファイル等にその詳細を記載します。

一般公開後のチェックリスト

以下は、ソース コードを一般公開した後に、ソース コード パッケージが入手可能になっているかを検証するチェックリストの例です。

- ソース コード パッケージは、ウェブサイトに問題なくアップロードされ、外部コンピューターからダウンロード可能になっています。
- ソース コード パッケージは、外部コンピューターでエラー無しに圧縮解凍できます。
- ソース コード パッケージは、外部コンピューターでエラー無しにコンパイルとビルドができます。

書面による申出

以下は、ソース コードを提供するための書面による申出の例です。

> この FooBar 製品で使用されているソフトウェアに関連して、FooBar 社(“FooBar”) によって一般に開示されているソース コードのコピーを入手するためには、書面で要求を以下に出すことが必要です。
>
> Foobar 株式会社
>
> Attention: Open Source Compliance Inquiries
>
> 住所
>
> 市、州、郵便番号
>
> 国名
>
> FooBar は、妥当な遅延の範囲で、http://opensource.foobar.com (“Website”)にてソース コードが入手可能になるように可能な全ての努力をいたします。書面による要求を出される前に、ソース コードが既に公開されているか、このウェブサイトを確認して下さい。

代替手段として、郵便ではなく email にて要求を受けたいならば、書面による通知の文言を少し変更します。

> この FooBar 製品で使用されているソフトウェアに関連して、FooBar 社(“FooBar”) によって一般に開示されているソース コードのコピーを入手するためには、書面で要求を opensourcecompliance@foobar.com に出す必要があります。
>
> FooBar は、妥当な遅延の範囲で、http://opensource.foobar.com (“Website”)にてソース コードが入手可能になるように可能な全ての努力をいたします。書面による要求を出される前に、ソース コードが既に公開されているか、このウェブサイトを確認して下さい。

第 6 章

コンプライアンス プロセス管理の推奨プラクティス

この章では、オープン ソースを商用製品に統合する際の、推奨プラクティスやさまざまな考察を扱います。3 つの部分に分かれます。

- 推奨プラクティスは、オープン ソース コンプライアンス管理の一貫プロセスの中のさまざまなステップに配置されます。
- ソース コード改変、告知、頒布、ソフトウェア設計、利用、リンク、コード結合など関連したコンプライアンスについての考察。
- 推奨プラクティスは、オープン ソース コンプライアンス プログラムの中のさまざまな不可欠要素に関連します。

コンプライアンス プロセス

コンプライアンス管理プロセスは、製品ソフトウェア スタックへの組み込みが承認される前に、ソフトウェア コンポーネントが通過するさまざまなステップを含んでいます。プロセスは、製品ビルド システムへ統合するさまざまなソフトウェア コンポーネントを特定することで始まり、結果として発生するライセンス義務をリスト化することで終わります。

以下の節では、コンプライアンス要求を処理するための推奨プラクティスを示します。推奨プラクティスは、図 24（次ページ）に示したコンプライアンス プロセスの中のステップに直接配置されます。

特定フェーズ

コンプライアンス プロセスの特定フェーズでは、組織は、ビルド システムへ入る全てのコンポーネント、その由来、ライセンス情報を特定します。入るソース コードには、主に 3 つの由来があります。

- 内部開発者によって作られたプロプライエタリ ソフトウェア。オープン ソースのコード断片を含んでいるかもしれません。コンポーネント レベルでオープン ソース コードと依存関係を持ったり、リンクされたりして、オープン ソースと統合されているかもしれません。
- 独立プロバイダーやコンサルタントによって開発され、商用ライセンスやオープン ソース ライセンスで許諾されたサード パーティ ソフトウェア。このソフトウェア カテゴリは上と同様にコード断片や依存関係を含んでいるかもしれません。
- オープン ソース プロジェクトのメンバーによって開発されたオープン ソース ソフトウェア。

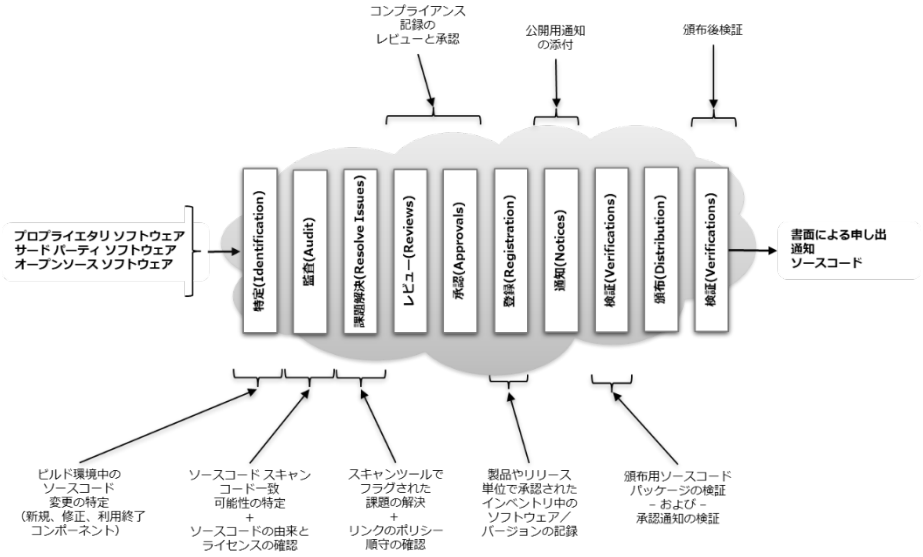


図 24：コンプライアンス一貫管理プロセス

入るソフトウェア コンポーネントは、全て特定され、コンプライアンス プロセスを通されることが推奨されます。

ソース コード監査

ソース コード監査やスキャンに関して、3 つの推奨プラクティスがあります。

全ソース コードスキャン

開発チームがオープン ソースをプロプライエタリ ソース コードやサード パーティ ソース コードに使用しているかもしれないので、製品やサービスに統合される全てのソース コードをスキャンします。さらに、開発チームは、オープン ソース コンポーネントに変更を入れているかもしれませんが、これは、追加の精査を必要とするかもしれませんし、潜在的に追加義務を負うかもしれません。したがって、製品に組み込まれている全てのソース コードを監査し、特定することが重要です。

前に承認されているパッケージの新しいバージョンのスキャン

ある場合には、前に承認されているパッケージが変更されて（同じ条件や異なる条件で）再利用されたり、そのまま再利用されたり、変更されて異なる製品やサービスに再利用されたり、新しいバージョンがダウンロードされてソフトウェア スタックに使用されたりします。コンプライアンスは、製品ごと、サービスごとに確認されますので、一つのケースでの利用承認は、必ずしも全てのケースに適用されることにはなりません。

規則として、開発者が前に承認されたコンポーネントを変更する時や、別の条件で前に承認されたコンポーネントの利用を計画する時には、その都度、ソース コードは再スキャンし直し、コンポーネントは再度承認プロセスを通過するべきです。

オープン ソース コンポーネントの各新しいバージョンが、レビューされ、承認されることを確実にする~~します。~~

バージョン更新の間に、ライセンス変更が起きる可能性があります。開発者がオープン ソース パッケージのバージョンを更新するときには、新しいバージョンに適用されているライセンスが、古いバージョンに適用されていたものと同じであることを確認します。

「早めのリリース、頻繁なリリース“release early and release often.”」というオープン ソースに関する格言があります。

オープン ソース開発モデルは、プロジェクト初日に始める頻繁なリリースを推奨しています。これによってユーザーに評価実験や不具合報告の機会を与えます。その目的は、品質保証活動が開発プロセスの定常的な一部となることです。

「早めに、そして頻繁にスキャン“Scan early and often”」は、同じ精神に従っています。開発プロセスにおいて早期にソース コードスキャンを行い、それを定常的に継続することは、コンプライアンスの取り組みが開発の取り組みに後れを取らないようにします。組織は、プロセスがより効率的になるように、新しいスキャンが必要とされる時を定義した条件リストを作成すべきです。

「早めに、そして頻繁にスキャン“scan early and often”」というアプローチには、いくつかの有利な点があります。

- それは、プロセスの早い段階でのコンプライアンス課題の発見を助けます。
- それは、出荷スケジュールへ深刻な問題を与えないような許容できる期間で、発見された問題へ解決をもたらすことを促進します。
- それは前に実行したソース コード スキャンの中でスキャンが必要なソース コード差分を減らすので、インクリメンタル スキャンの実行効率を改善します。

課題解決

ソース コードがスキャンされて、コンプライアンス課題が発見され、フラグがつけられた時、課題解決にはいくつもの方法があります。

- スキャン結果に疑問がある場合は、技術担当者と検討します。（問題となっているソフトウェア コンポーネントの担当開発者にインタビューします。）
- スキャン ツールでフラグを付けられた各ファイルやコード断片を精査し、解決します。驚くような由来のソース コードと一致することがあります。
- オープン ソースに施された変更を特定します。理想的には、コード変更をしたかどうか、（ドキュメントはいうに及ばず、）技術者の記録に依存するべきではありません。誰が、いつ、コード変更を行ったかを特定するには、ビルド ツール（SCM、build automation、etc.）に頼るべきです。

- 例えば、ソース コード スキャン ツールが、プロプライエタリ コンポーネントの中に未承認の GPL 許諾されたソース コード（コード断片）の利用を発見したならば、このことについて、技術担当に訂正要求を提出します。技術担当が課題を解決した後、問題のソース コードが削除され、適切で同等のコードによって置き換えられていることを確認するために、ソース コードを再スキャンすることを推奨します。
- 法務レビューの準備としては、特定のコンポーネントに対して発見されたライセンス情報の全てを、法務担当に提供するのが最も良い方法です。
    - スキャン ツールによって生成されたソース コード監査報告
    - オープン ソース コンポーネントに対する COPYING、README、LICENSE ファイル
    - サード パーティ ソフトウェア プロバイダーから受け取るソフトウェア コンポーネントに対するライセンス契約

レビュー

コンプライアンス プロセスの一部として行われる、異なるタイプのレビューがあります。この節では、アーキテクチャー レビューとリンク解析レビューを議論します。

アーキテクチャー レビューは、オープン ソースとプロプライエタリやサード パーティ ソフトウェア コンポーネントとの相互作用に関する解析です。企業は、しばしば、問題となっている製品に責任を持つアーキテクトと、さまざまな重要ソフトウェア コンポーネントに責任を持つ開発者を参加させて、アーキテクチャー レビューを実施します。

このレビューの目的は、特定することです。

- オープン ソース コンポーネント（そのまま利用、または変更利用）プロプライエタリ コンポーネント商用ライセンスで許諾されたサード パーティ コンポーネント
- コンポーネントの依存状態
- コンポーネントとサブシステム間の通信プロトコル
- 動的リンクと静的リンク（以下の節で議論されます）
- カーネル空間実行（ドライバ等）とユーザー空間実行（ライブラリ、ミドルウェア、アプリケーション）
- 共有されたヘッダファイルを使用するコンポーネント
- 特に別のオープン ソース ライセンスで許諾されているような場合、特定のソフトウェア コンポーネントと相互作用する、または依存関係にある、別のオープン ソース

アーキテクチャー レビューの結果は、オープン ソースからプロプライエタリ コンポーネントやサード パーティ コンポーネントへ影響するかもしれないライセンス義務の分析です。

承認

コンプライアンス プロセスの承認段階の一部として、2 つの推奨プラクティスがあります。

- コンプライアンス チケットに関連する全てのサブタスクは、コンプライアンス チケットを承認する前に、完了し、クローズされていることを検証します。サブタスクや保留中の小課題は簡単に忘れてしまいますが、そうすると、未解決の課題が残っている時でさえもコンプライアンス チケットをクローズするような未熟なプロセスに陥ってしまいます。
- 承認や却下の判断を下した議論の概要を記録します。そうした文書は、該当するコンポーネントに提示された承認の基礎が何であったのか、課題はどのように解決されたのか、を特定するときに非常に役に立ちます。

通知

製品やサービスにオープン ソースを利用している組織は、以下が必要となります。

- 著作権や帰属を全て通知することで、オープン ソースを利用していることを知らせます。
- （例えば、GPL、LGPL で許諾されたソース コードのように、適用可能な場合には）エンド ユーザーにオープン ソース コードのコピーの入手方法を伝えます。
- 製品に含まれているオープン ソース コードに対応するライセンスの全文を提示します。

この分野のいくつかの推奨プラクティスには以下のものがあります。

- オープン ソースの利用が承認されるその都度、帰属とライセンス文を追加的に集めます。この方法に従いますと、要求される通知ファイルは、常に最新に更新され、全てのオープン ソースのリスト、ライセンス情報、著作権、帰属通知を含むことになります。
- 書面による申出には、明快な言葉を使い、製品に使われている全てのオープン ソースが含まれるようにします。
- 製品のエンド ユーザーが、この情報が、製品そのものの中、製品文書（ユーザーマニュアルまたは CD-ROM）の中、かつ／またはウェブサイトなど、どこに置かれているかを知ることができるようにします。

検証

コンプライアンス チームが、一貫性を確認して、検証ステップが見落とされていないかを確認する時に従うチェックリストを、作成し維持し発展させるのは大変有益で効率的なことです。頒布前検証の例としては以下があります。

- 頒布されることになっているオープン ソース パッケージは、特定され、承認されている。
- 不適切なコメントは、ソース コード パッケージから取り除かれている。（厳密には、これはコンプライアンス課題ではありません。しかしながら、コメントは、見えていないコンプライアンス課題を明らかにするかもしれません。）
- ソース コード パッケージ（変更分も含めて）は、製品やソフトウェア スタックに入って出荷されるバイナリと一致するものが、入手可能にされる。
- エンド ユーザーにオープン ソースのソース コードを要求する権利を伝える文書通知に加えて、適切な通知が、製品文書に入れられる。

オープン ソース パッケージが頒布用ウェブサイト（かつ／または、同等のメディアに格納）にアップロードされたとしても、作業は完了していません。以下を検証する必要があります。

- パッケージは、正しくアップロードされている。
- パッケージは、外部コンピューターでエラー無しにダウンロードして圧縮解凍できる。
- 含まれるパッケージは、適切にコンパイル、ビルドできる。
- 開発者は、将来の製品、製品開発コード名、競合他社への言及、その他不適切なコメントを残していない。

ツールと自動化

ツールは、組織がコンプライアンス活動を効率的で正確に実行するのを助ける、コンプライアンス プログラムの中の不可欠要素です。多くのツールは、オープン ソース コンプライアンス プログラムの中で、とても有用であることを証明しています。

- ソース コード スキャンとライセンス特定ツール
- プロジェクト管理ツール
- BoM 比較ツール
- リンク解析ツール

以下の小節では、ツールに関する情報と、ツール利用がコンプライアンス活動にどのように活用できるかを示します。市場では、以下で記述するようなさまざまな機能を提供する、複数の商用、プロプライエタリ、オープン ソースのツールがあります。

ソース コード特定ツール

ソース コードとライセンスを特定するツールは、ユーザーがオープン ソース ソフトウェア コンポーネントに関連するソース コードとライセンスの由来を特定するのを助けるような、検出と解析の機能を提供します。

- Antelink Reporter: http://www.antelink.com/
- Black Duck Protex: https://www.blackducksoftware.com/products/protex
- The Black Duck Hub: https://www.blackducksoftware.com/products/hub
- FOSSology: http://www.fossology.org/projects/fossology
- nexB DejaCode: http://www.nexb.com/products.html
- Open Logic Exchange: http://www.openlogic.com/products-services/openlogic-exchange
- Palamida Enterprise: http://www.palamida.com/products/enterprise
- Protecode Enterprise: http://www.protecode.com/our-products/
- WhiteSource: http://www.whitesourcesoftware.com

プロジェクト管理ツール

プロジェクト管理ツールは、コンプライアンス活動を管理し追跡するのに不可欠なものです。いくつかの企業では、カスタマイズしたコンプライアンス ワークフローと共にバグ追跡ツールをすでに実際に利用しています。他の企業では、特定のプロジェクト管理ツールや自社開発ツールを利用しています。どちらにしても、ツールは、プロセスのある段階から別の段階へコンプライアンス チケットを移し、タスクやリソースの管理、時間追跡、email 通知、プロジェクト統計、報告機能などを提供して、コンプライアンス プロセスのワークフローを反映させるべきです。

コンプライアンスで広く利用される不具合追跡ツールの例

- Bugzilla: https://www.bugzilla.org/
- IBM Rationale ClearQuest: http://www.ibm.com/software/products/en/clearquest/
- JIRA: https://www.atlassian.com/software/jira
- Redmine: http://www.redmine.org/
- Bugzilla: https://www.bugzilla.org/

ソフトウェア BOM 差分ツール

ソフトウェア BOM 差分ツールの目的は、二つの BOM の差分を計算し、変更リストを生成することです。このようなツールは、（例えば、リリース 1.1 から 1.2 へ移行するなど）既存のベース コードの新しいバージョンを扱う時、インクリメンタル コンプライアンスを効率的に実施するのを可能にします。BOM 差分ツールへの入力は、製品やサービスのコード ベースの二つのバージョンのコンポーネント リストを示す二つの BOM ファイルです。BOM 差分ツールの出力は、新しいコンポーネント、使わなくなったコンポーネント、および変更されたコンポーネントのリストです。

BOM 管理ツールは、物理的な製造の世界には多くのものがありますが、オープン ソース ソフトウェアの管理用途のものは少ないです。著者の経験では、オープン ソース管理プロセスを支援する BOM 差分ツールは通常、自社内製であるか、かつ／または、既存ツールや機能をより集めて作られたものです。BOM のフォーマットにも依存しますが、ウェブベースで BOM バージョン比較を作成するために、コマンドラインの diff ツ

ールや、生産性ツール(spreadsheets 等)や、ディレクトリ比較ツール、ビルドや連続統合ツールからの報告に加えて、これらを結合するスクリプトを利用することが可能です。図 25 は、図での説明目的のために作られてものですが、自社内製の BOM 差分ツールのサンプル出力を示しています。

BOM Difference Report

Package	Image	Owner	Version	Submission
adapterbase	none	chad.everett@acme.co.nz > richardburton@acme.co.nz	1.0.0	23
adec-omxump3-msm7x25	customization	pamela.anderson@acme.co.nz	1.0.0	1 > 2
amazonservice	luna	una.due@acme.co.nz	1.0.0	4 > 5
audiod	luna	damien.lewis@acme.co.nz	1.0	111 > 114
audiod-config	luna	damien.lewis@acme.co.nz	1.0	70 > 72
browser-adapter	cust	chris.reeve@acme.co.nz	1.0.01	118 > 131
browser-service	cust	chris.reeve@acme.co.nz	1.0.01	126 > 128
bzip2	rockhopper	> eric.idle@acme.co.nz	1.0.2	1
camd-omap	any	itzaak.periman@acme.co.nz	1.0.0	5 > 7
cifs	rockhopper	> eric.idle@acme.co.nz	3.0.23c	1
clam	any	sean.connery@acme.co.nz	1.0.0	7 > 8

図 25：BOM 差分報告の例

リンク解析ツール

依存関係解析ツールの目的は、C や C++言語に特有の話題ですが、動的リンク、静的リンクのレベルで問題のあるコードの組み合わせにフラグを付けることです。このツールは、ツールの使用者が事前に定義したライセンス ポリシーに基づいて、バイナリとライブラリのライセンス間のリンクの矛盾を特定します。

上記のソース コード特定ツール、多くの静的解析ツール、ビルドや統合の報告ツール、自社内製のコード解析ツールなど、依存関係を解析するために一緒に利用されるツールが多くあります。依存関係マッピングに対する主な要求事項は、以下の機能です。

- バイナリとライブラリ間のリンクを特定します。
- バイナリとライブラリのライセンスを特定します。
- ライセンス スキャン ツールと接続します。
- （GPL で許諾されたコードとのリンクなど）方針から逸脱しているリンクにフラグを付け、企業方針と合うように構成します。

著者の経験では、依存関係マッピング ツールは、BOM ツールとよく似ていますが、通常、自社内製であるか、かつ／または、既存ツールや機能をより集めて作られたものです。

このタイプのオープン ソースの既存ツールとしては、Linux Foundation の Dep-Checker があります (http://git.linuxfoundation.org/dep-checker.git/)。

第 7 章

コンプライアンスに関する照会の管理

この章は、コンプライアンスに関する照会を扱うガイドラインについて示します。これらのガイドラインは、申し立てを調査している間に申立人に対して肯定的で協力的な態度を維持することと、ライセンス違反が実際に起きた時には適切な行動を取ることを、確実にします。

コンプライアンス情報の提供要求を無視した後に、否定的な評判を受けたり、かつ／または、法的手続きを受けた組織、コンプライアンスに関する照会をどう扱えばよいかわからなかった組織、コンプライアンス プログラムを持っていなかったり整備していない組織、ライセンスが強制的でないと（誤って）考えて協力を単に拒む組織があります。今日、ベスト プラクティスは、これらのアプローチがいずれの関係者の利益にもならないことを教えてくれます。したがって、企業は、コンプライアンスに関する照会を無視するべきではありません。むしろ、照会を受け取ったことを知らせ、照会者に回答を保留していることを伝え、回答予定日を知らせるべきです。

コンプライアンスに関する照会は、以下の要求も含まれます。

- GPL、LGPL やその他のライセンスで許諾されたソース コードを提供するという文書通知に従って行われる、ソース コードへのアクセス
- 製品内にあることがわかったが、開示されていないコンポーネントに関するソース コードへのアクセス
- 特定のオープン ソース コンポーネントが製品やサービスで利用されているかどうかの確認
- 無効となった帰属や著作権の通知に対する更新
- ライセンス義務の一環として入手可能になっているオープン ソース パッケージにおいて、抜けているファイルの提供

企業は、通常、コンプライアンスに関する照会を、書面による申出やオープン ソース通知の一部として案内している専用 email アドレス経由で受け取ります。

コンプライアンスに関する照会への対応

この節では、コンプライアンスに関する照会への対応方法について紹介します。図 26 は、照会を受け取ってからクローズするまでの各段階を説明する、コンプライアンスに関する照会への対応プロセス例を提示します。

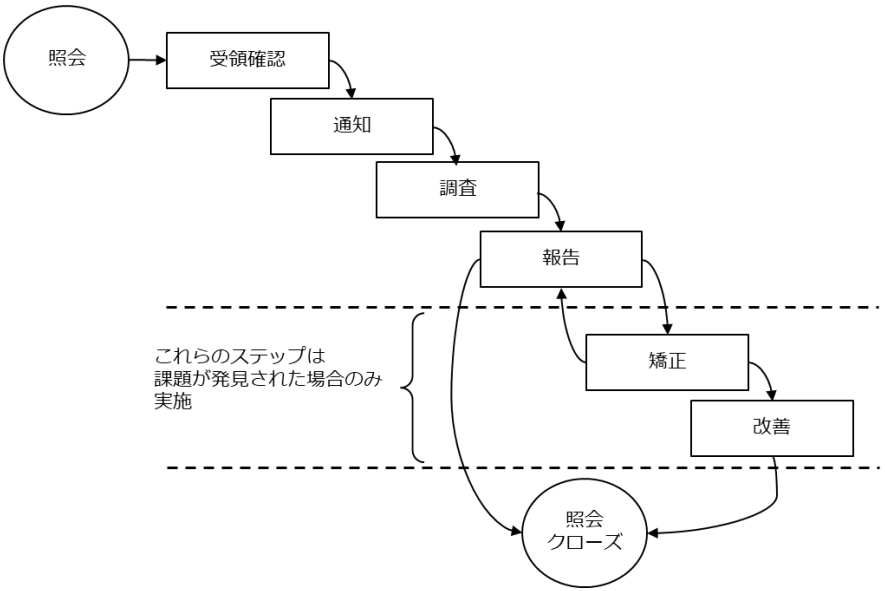


図 26：コンプライアンスに関する照会へ対応するプロセス

確認

コンプライアンスに関する照会を受け取ったら、照会を受け取ったことを伝え、期日を定めて調査を約束するなどの応答をすぐに行うべきです。

照会者の身元と動機を理解し、主張が正当で正確なものでその時点で意味のあるものかを確認するのは重要です。照会者がライセンスを十分に理解しておらず、誤った仮定で照会を提出することがしばしばあることを理解します。照会が情報を欠いている時には、追加で次のような確認を要求します。

- 影響のある製品やサービスや、該当するコードの名前
- 違反があると信じる理由
- プロジェクト コードと違反状態にあるライセンスの名前
- プロジェクト サイトへのリンク

通知

照会者との率直な対話を維持することが推奨されます。常に、自分たちのオープン ソース コンプライアンス プラクティスを示し、コンプライアンスに対する長年の誠実な取り組みを行動によって示します。照会者に自社のコンプライアンス手続きとプラクティスを伝え、照会された事項を調査することを確約します。提示可能になったところで、内部調査に関する最新情報を伝えることは良いことです。

調査

この段階では、報告済みの申し立て案件を調査して、問題となっているコンポーネントに関するコンプライアンス記録を参照し、レビューし、コンプライアンス記録と照会とを比較するべきです。

報告

許容できる期間内で内部調査の結論を出して発見された事項を内部記録に残した後、照会者に結果を伝える必要があります。

照会のクローズ

コンプライアンスに関する照会が誤った警告であった場合には、（照会者に結論を伝える以外には）特に何もしないでコンプライアンス チケットをクローズすることができます。

矯正

調査がコンプライアンス課題を明らかにした場合には、照会者に対し、製品やサービスをコンプライアンス遵守の状態に戻すために必要な手順を全て実行することと、これらの作業を完了するのに予想される日程を明確にして、事実を伝えるべきです。協力的で誠意をもって照会者と一緒に課題を解決するのが、担当者の責任です。適用されているライセンスに基づく義務を理解していることを示し、いつまでにどのようにして義務を満足するかを伝える必要があります。

問題を解決したら、紹介者にすぐに伝え、解決策を検証するために来てもらうべきです。

改善

コンプライアンス課題があった場合には、OSRB を招集して、事例を議論し、どのようにしてこのコンプライアンス違反が生じたかを学び、そのような誤りが再び生じないように、既存のプロセスとプラクティスを改善するべきです。

一般的な考察

- 全ての照会は、正式な照会として扱います。照会者とのやりとりの中で開示する情報は、一般に開示する可能性があるという前提で作業をします。
- オープン ソース コンプライアンスに対して努力することは、ライセンス強制を受けた時にいかに役立ち、自社プロセスをいかに改善するかを考えてみて下さい。

第 8 章

その他のコンプライアンス関連プラクティス

この章は、実際のコンプライアンス プロセスの範囲外ではありますが、コンプライアンスのベスト プラクティスやさまざまな考察に光を当てます。

従業員の評価

技術面やコンプライアンス強制に関連して、全ての企業が直面する 4 つの挑戦があります。

- 技術者が利用したいオープン ソース コンポーネント毎に一貫して要求フォームを確実に記入するようにします。
- コンプライアンス チケットに対して適時に応答するように、技術者に要求します。
- OSRB によって定められたガイドラインに従って技術者が行動していることを検証します。
- 組織内のオープン ソース コンプライアンスに関するトレーニングを技術者が受けるように義務付けます。

これらの 4 つの挑戦に直面した企業をサポートするのに効果的であったプラクティスは、従業員の実績評価の一部としてオープン ソースとコンプライアンス評価項目を含めることです。結果として、従業員のボーナスの一部は、コンプライアンス ポリシーと手順に彼らが従った程度によって決まります。レビューは、従業員が以下の項目を実行したかを評価します。

- 利用するオープン ソース コンポーネント毎に OSRB フォームを記入する。
- Respond to compliance tickets without significant delays
- マネージャによって設定された期限内にオープン ソースとコンプライアンスに関するトレーニングを修了する。
- OSRB が制定したガイドラインの範囲内でオープン ソースを利用し、コンプライアンス違反を起こさない。

逆に、コンプライアンスを従業員の実績評価として利用するためには、OSRB は各担当者の以下の点を追跡しなければなりません。

- ソフトウェア BOM に含まれているが、承認を受けていないコンポーネント
- コンプライアンス チケットへの応答時間
- トレーニングコース修了
- 経営チームへ報告されたコンプライアンス違反

ウェブ ポータル

いくつかの企業は、社内向け、社外向けのオープン ソース ウェブ ポータルを運営しています。社内向けポータルは、コンプライアンス ポリシーやガイドライン、トレーニング教材、通知、関連するメーリング リストなどを提供しています。社外向けポータルは、ライセンス義務を履行する形で、利用しているオープン ソース パッケージのソース コードを一貫した方法で提供します。

意思伝達

意思の伝達に関する最も重要で唯一の推奨事項は、社内にオープン ソースに関連する企業目標や懸念事項を説明するか、社外のコミュニティ参加者に対するかに関わらず、明確で一貫性を持つことです。コミュニティとの接点となるサイトを持つことは、コンプライアンスに関する照会に応答する時に特に重要になります。

トレーニング

オープン ソースとコンプライアンスに関するトレーニングの目標は、オープン ソース ポリシーと戦略についての意識を高め、オープン ソース ライセンスの課題と事実について共通理解を形成することです。トレーニングは、製品にオープン ソースを統合することによるビジネスリスク、法的リスクも扱います。また、組織のコンプライアンス ポリシーやプロセスを広めて推進し、コンプライアンス文化を促進する手段でもあります。

公式と非公式のトレーニング方法があります。公式の方法は、修了するために従業員は試験に合格する必要がある、インストラクターが担当するトレーニングコースを含みます。非公式の方法は、ウェブでのトレーニング、ブラウン バッグ セミナー、新規雇用従業員へのオリエンテーションの一部としての説明などを含みます。

非公式トレーニング

ブラウン バッグ セミナー

ブラウン バッグ セミナーは、企業従業員（法務担当者、オープン ソース エキスパート、コンプライアンス オフィサー等）や招待スピーカ（広く行われているのは著名なオープン ソース開発者）による、昼食時の講演会です。これらのセミナーの目標は、製品やソフトウェア スタックに統合されているオープン ソースのさまざまな面に関して説明して議論を引き出すことです。これらのセッションは、企業のコンプライアンス プログラム、ポリシー、そしてプロセスに関する議論も含みます。

新規従業員へのオリエンテーション

いくつかの例では、コンプライアンス オフィサーが、企業のコンプライアンスに関する努力や規則、ポリシー、そしてプロセスをオリエンテーションの一部として新規従業員に説明します。初日に、新規従業員は、オープン ソースとコンプライアンスに関する 30 分のトレーニングを受けます。結果として、新規従業員は、社内で誰がその内容のエキスパートであるか、どういう社内イントラネットがあるか、オープン ソースとコンプライアンスに関するトレーニングがどのように登録されるかなど、必要な情報を受け取ります。

公式トレーニング

組織の規模や、オープン ソースが商品に使われている範囲にもよりますが、組織は、オープン ソースに携わる従業員が公式な講師の教えるコースを受講し、その領域での習熟度のテストを受けるように命じることができます。

ソース コード変更に関する考察

既存のソース コードを変更する基本ルールを確立するために、平易で法務用語を使わない言葉で表現した、内部利用目的のガイドラインを発行することを強く勧めます。
例

- プロプライエタリを残すソース コード変更は、特に派生物への義務を持つような（GPL、LGPL 等の）オープン ソース パッケージ内で行っていはいけません。
- プロプライエタリ ソース コードは、派生物への義務を持つようなオープン ソース ライブラリにリンクしてはいけません。企業では、通常、それらを実行するには正式な OSRB 承認を必要とします。
- ソース コードに加えられるいかなる変更も、オープン ソース ライセンスに適合して頒布前に文書化されることを確実にします。
- オープン ソース モジュールへの全ての変更は、モジュールの修正履歴（変更ログファイル）で把握されるようにします。

通知に関する考察

オープン ソースを使う時に重要な義務の一つは、著作権、帰属、ライセンス情報、そして書面による申出（GPL/LGPL で許諾されたソース コード）などの明確で正確な文書作成を確実にすることです。これらの文書化の義務の全体は、しばしばオープン ソース通知と呼ばれます。

提供物にオープン ソースを利用している企業は、著作権の帰属を全て表示し、ほとんどの場合、製品やサービスに含まれているオープン ソース ソフトウェアのライセンス全文を記載することで、オープン ソース利用を知らせなければなりません。したがって、企業は、出荷する全ての製品や提供する全てのサービスの文書に、著作権、帰属、そしてライセンス通知を記載することで、文書化の義務を満たさなければなりません。

文書化義務の要求を満たす 2 つの主要な選択肢があります。

- 製品自体にオープン ソース通知を表示します。これは、ユーザーと対話してライセンス情報を引き出すか、ライセンス情報を表示するかできるユーザー インターフェイスを製品が持っている場合には、実行可能な選択肢です。この選択肢の例は、携帯電話やタブレットです。
- 製品マニュアルや、製品に付属する文書に、オープン ソース通知を含めます。

いくつかの企業は、ウェブサイト（一つの選択肢ではありますが、頻繁に採用され、維持にも手がかかりません。基本的にはウェブサイト上に通知ファイルを置くだけです。）上で通知を維持しながら、可能な時には両方の選択肢を選びます。通知に関する考察から得られる重要なことは、製品出荷やサービス開始前に、全てのオープン ソース通知に対する要求が満たされることを、確実にしなければならないということです。

頒布に関する考察

一般的に言って、企業は、製品出荷前にオープン ソース頒布に関する義務を果たした状態にできることを望みます。コンプライアンス プラクティスを開発サイクルで徹底的に実施することによって、頒布に関する考察はとても単純で簡素なものになります。

利用に関する考察

以下の節では、完全にコンプライアンスに適合したオープン ソース利用についての、考察と警告を示します。

BoM のクリーン化

入るソフトウェアが、利用宣言されていないオープン ソースを含んでいないことを確実にします。供給者から受け取る時に、いつもソース コードを監査します。代わりの方法としては、ソフトウェア供給者が供給するコードの監査報告を必ず提供しなければならないという企業ポリシーを制定します。

各オープン ソース コンポーネントに対する OSRB フォーム

利用するそれぞれのオープン ソース コンポーネントに対して OSRB 利用要求フォームを記入します。明白な OSRB 許可がないオープン ソースを利用するのは避けます。

M&A によるリスクの理解

利用されているオープン ソース コードと関連する事項は、企業間の取引に先立って実施される資産監査対象の一部であることを理解します。

使用しなくなったオープン ソース パッケージ

承認済みのオープン ソース パッケージが利用されなくなったならば、技術者は OSRB に連絡してオープン ソースリストを更新するように通知しなければなりません。さもなければ、OSRB が、BOM 差分ツールを実行する際にもはや使われなくなっているパッケージを見つけるでしょう。

大きなソース コード変更

承認済みのパッケージが大きな変更を受けた時には、OSRB にソース コードを再スキャンするように通知します。さもなければ、OSRB が BOM 差分ツールを実行する際にパッケージが変更されていることを発見するでしょう。設計や実装での大きな変更は、アーキテクチャー、API、ユースケースにインパクトを与えますし、ある場合にはコンプライアンス面でもインパクトがあります。

参考オリジナル ソース コード

ダウンロードしたパッケージを保存するのに加えて、オープン ソース パッケージをダウンロードした URL も文書化します。

オープン ソースの新しいバージョンへの更新

同じオープン ソース コンポーネントの新しいバージョンがレビューされ承認されることを確実にします。バージョン更新の間でライセンス変更が起きることがありますので、オープン ソース パッケージのバージョンを更新する時には、新しいバージョンのライセンスが前のバージョンから変更されていないことを確認します。もしライセンスが変わっていたら、コンプライアンス記録を更新して、新しいライセンスが問題を起こさないように、OSRB に相談します。

コンプライアンス 検証の黄金律

コンプライアンスは製品ごとサービスごとに検証されます。というのは、オープン ソース パッケージは、ある条件での利用を承認されたのであり、2 次利用について承認されたわけではないからです。

コピー／ペースト

事前の OSRB 承認なしでの、ソース コード断片の利用や、オープン ソース コードをプロプライエタリやサード パーティ ソース コードへ（その逆も）コピー／ペーストすることを避けます。これらの行動は、コンプライアンスに深く関わります。

ソース コードの異なるライセンスとの結合

多くのオープン ソース ライセンスは互いに両立性を持っていないので、派生物に異なるオープン ソース ライセンスを結合することを避けます。このトピックに関して、法務担当からのサポートを受けることを強く勧めます。

ソース コード コメント

ソース コードに不適切なコメントを残さないようにします。（個人的なコメント、商品コード名、競合他社名等）

既存ライセンス情報

既存の著作権やライセンスの情報をオープン ソース コンポーネントから削除あるいは変更しないようにします。全ての著作権やライセンスの情報は、ライセンスが変更を許していると完全に確信できるとき以外は、オープン ソース コンポーネントでそのままの状態にしておきます。

帰属についての考察

製品にオープン ソースを組み込んでいる企業は、エンド ユーザーに帰属情報を提供する必要があります。この節では、オープン ソースの帰属に関する義務をどのように達成するかについてのガイドラインを示します。

帰属タイプ

オープン ソースの帰属要求は、ライセンスによって異なりますが、一般的に 4 つのカテゴリに分けられます。

ライセンス全文

ライセンス全文の逐語的コピーが、ほとんど全てのオープン ソース ライセンスで要求されます。

著作権表記

著作権表記の逐語的コピーが、多くのオープン ソース ライセンスで要求されます。

謝辞表記

いくつかのオープン ソース ライセンスは、明示的に著者の帰属を要求します。ほとんどの場合、オープン ソース プロジェクトは、貢献者のリストを含む AUTHORS と呼ばれるファイルを含んでいますので、帰属表記の一部としてこの情報を使うことができます。

ソース コード入手についての情報

ソース コード再頒布義務を持つほとんどのライセンスは、製品に付随するソース コード提供か、ソース コードの入手方法を記述した書面による申出かを要求します。GPL と LGPL は、このカテゴリのライセンスの例です。

帰属の提示

オープン ソースを含むあるいは利用している製品やサービスに関していうと、帰属は、（製品マニュアル等の）印刷されたユーザー文書や、CD またはウェブサイトからのダウンロードなど、印刷または電子形式で頒布されなくてはなりません。もし、製品やサービスがグラフィカル ユーザー インターフェイスやコマンド ライン インターフェイスを持っているならば、帰属をその UI 経由で表示することも選択肢の一つです。携帯電話での通信での（OTA）製品アップデートに関していうと、製品アップデートが新しいあるいは更新されたオープン ソース コンポーネントを含むのであれば、帰属は、更新されなければなりません。

特定のライセンス義務

「エンド ユーザーが利用可能な文書にライセンスのコピーを入れなければならない」

問題となっているオープン ソース コンポーネントのライセンスは、このオープン ソースを利用する全ての製品のユーザー文書に含まれていなければなりません。

推奨事項

- 携帯電話やタブレットなど、いくつかの例では、製造会社は、ウェブ ブラウザや PDF ビューアを使って（ライセンス文は、HTML や PDF 形式で機器上で利用可能です）、実際の機器上で通知を提示することができます。
- ユーザーがアクセス可能なファイルシステムを有する製品では、ライセンスは、意味が分かるようするために、また、オープン ソースのライセンス ファイル名と同じようにするために、LICENSE というファイル名でファイルシステムに含まれていることを推奨します。

- 製品アップデートに関して、ライセンス情報もまたアップデートされなければなりません。例えば、新しいソフトウェア リリースが入手可能になった時、アップデートされたリリースは、新しいリリースで導入されたオープン ソースの全ての変更を反映している、アップデートされたライセンス情報ファイルを含んでいなければなりません。変更は、以下を含むかもしれません：
  - 新しいオープン ソース
  - 評価が下がった／使われなくなったオープン ソース新しいバージョンにアップグレードされたオープン ソース。
  - 帰属／著作権表記の更新を必要とするかもしれません。まれにライセンスの更新を必要とすることもあります。

「エンド ユーザーが利用可能な文書に著作権表記を入れなければならない」

問題となっているオープン ソース コンポーネントのライセンスは、エンド ユーザーが利用可能な製品文書中に著作権表記を必要とするかもしれません。

推奨事項

- 全ての製品に関して、著作権情報は、（ユーザーマニュアルのような）印刷物に含まれていなければなりません。
- グラフィカル ユーザー インターフェイスを有する場合には、エンド ユーザーは「製品について」や「ライセンス」の画面から著作権情報を参照することができるようにすべきです。
- 製品が、ユーザーが利用可能なファイル システムを有している場合には、著作権情報は、例えば製品で利用されている全オープン ソースに対する全著作権表記は、ファイル システムに含まれているべきです。
- 製品アップデートに関して、著作権情報もまたアップデートされなければなりません。

「広告資料は、特別な謝辞が必要になるかもしれません。」

オリジナルの BSD ライセンスに由来する広告文は、以下のように書かれています。

> All advertising materials mentioning features or use of this software must display the following acknowledgement: This product includes software developed by the University of California, Berkeley and its contributors.
>
> Where applicable, all marketing and advertising material (including web-based, magazines, newspapers, flyers, etc.) must display the acknowledgement.

一般的なガイドライン

オープン ソース プロジェクト名を推奨に利用しない、自分たちが行ったソース コード変更に印をつける、オリジナルのライセンス、著作権、帰属情報を保持する、などオープン ソース ライセンスを応用するガイドラインのいくつかについては既に良く知っているかもしれません。以下の節では、これらのガイドラインの細かな議論を発展させます。

推奨や推進に利用しない

オープン ソース プロジェクト、著者、貢献者などの名前を、事前に文書による許諾を受けないで、マーケティング、広告や文書（紙媒体、電子媒体、ウェブなど）に利用することはできません。

ソース コード変更の印

変更したオープン ソース コードを再頒布するとき、既存の著作権行をそのまま保持しながら、自分の変更箇所へは、著作権行（会社、年）によって印をつける必要があります。

ある会社は別のアプローチを取りました。オリジナルのオープン ソース コードと共に、そのコードに適用させる自社の貢献修正ファイルを提供しました。このアプローチに従えば、会社の修正は、オリジナル ソース コードと明らかに分離されます。

オリジナルのライセンス、著作権、帰属情報を維持する

オープン ソース コードを再頒布する時にはいつでも、変更が伴っても伴わなくても、オリジナルのライセンス情報、著作権行、やその他の帰属情報を維持しなければいけません。

ソース コード コメント

個人的なコメント、製品コード名、競合社への言及など、不適切なコメントをソースコードに残してはいけません。

既存ライセンス情報

既存のオープン ソース ライセンス著作権や他のライセンス情報を、利用するオープン ソース コンポーネントから取り除いたり、他のいかなる方法でも手を入れてはいけません。全ての著作権とライセンス情報は、全てのオープン ソース コンポーネントの中で触らずそのまま残しておく必要があります。

第 9 章

オープン ソース法務サポートを拡大させる

オープン ソース コンプライアンスは、しばしば、法務的な問題というよりも運用や物流的な問題です。コンプライアンスを達成するためには、適切なポリシーやプロセス、トレーニング、ツール、著作権者の著作権を尊重し、ライセンス義務を遵守し、組織や顧客や供給者の知的財産を守りながら、オープン ソースを効果的に利用しオープン ソース プロジェクトやコミュニティに貢献する体制が構築できる適切な担当者の配置などが必要です。

しかしながら、法務担当は、オープン ソース コンプライアンス プログラムや適切なコンプライアンスを確実にするもっとも重要なコア チームをサポートする不可欠な役割を果たします。この章では、オープン ソース コンプライアンスを確実にする法務担当の役割を詳細に見て、法務担当がソフトウェア開発チームに与えることができる実際的なアドバイスを提供します。そういう実際的なアドバイスは、ソフトウェア開発者が、法務担当に頼らなくてもできる、細かな問題に対する毎日の判断を可能にします。

実際的な法務アドバイス

法務担当からソフトウェア開発者に向けた実際的なアドバイスは、以下を含むかもしれません

- **ライセンス プレーブック：**ソフトウェア開発者を意図して書かれた、オープン ソース ライセンスについての簡単に読める要約形式のまとめ
- **ライセンス両立性マトリックス：**ライセンス A がライセンス B と両立性があるか判断するのを助ける表ソフトウェア開発者は、異なるプロジェクトの異なるライセンスにあるコードをマージする際に、そのようなマトリックスを使うことができます。
- **ライセンス クラス：**異なるライセンスと、これらのライセンスで許諾されたソース コードを利用する際に必要な行動を理解する簡単な方法

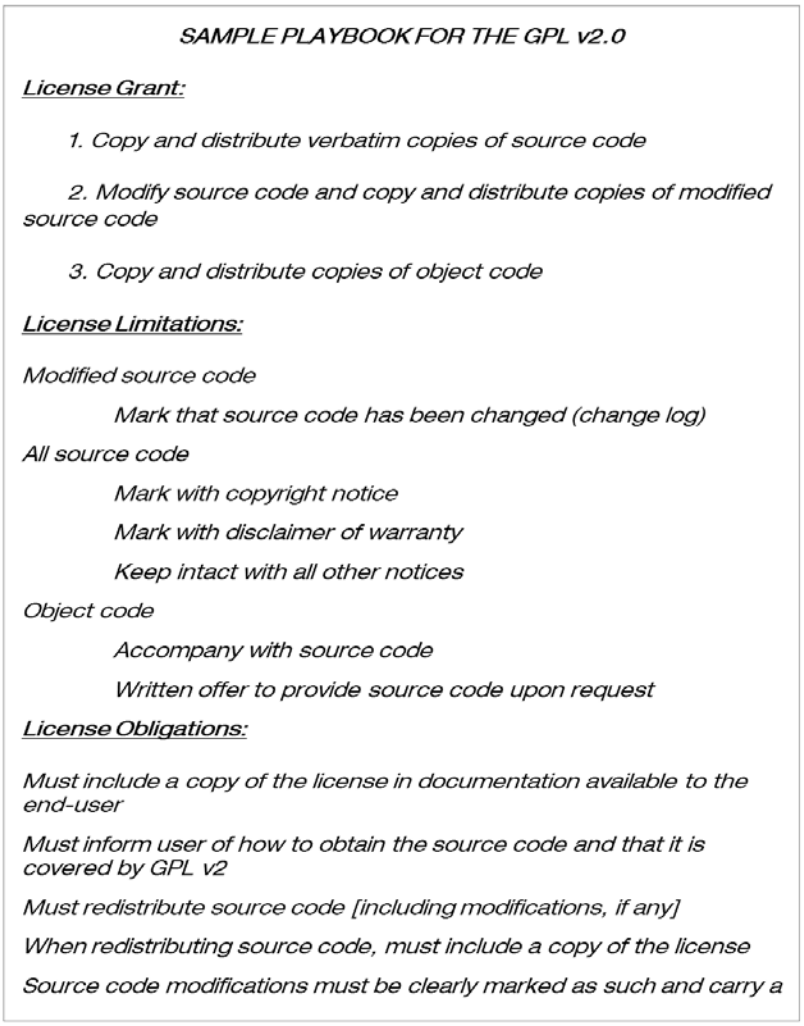
- **ソフトウェア相互作用法：**異なるライセンスで利用可能なソフトウェア コンポーネントがどのように相互作用するか、相互作用の方法は企業のコンプライアンス ポリシーで許可されているかを理解するガイド
- **チェックリスト：**開発やコンプライアンスのプロセスの全てのポイントで必要なことは何かを覚えておくための、一貫性のある、極めて簡単な方法

以下の節では、これらの 5 つのアドバイスを確認し、例を示し、これらによって開発者がオープン ソースに関わる時にどのように助けになるかを議論します。

ライセンス プレーブック

ライセンス プレーブックは、広く使われているオープン ソースについての要約です。ライセンスの許諾、制約、義務、特許への影響、その他、このライセンスに関する簡単に理解できる情報が書かれています。ライセンス プレーブックは、法務担当へ来る基本的な質問の数を最小化し、これらのライセンスに関する法務的な情報を即座に開発者に提供します。

図 27（次ページ）は、GPL v2 に関するライセンス プレーブックの例を示します。このプレーブックは、説明目的で書かれたものであり、その中身は確実と考えないように注意して下さい。



SAMPLE PLAYBOOK FOR THE GPL v2.0

License Grant:

1. Copy and distribute verbatim copies of source code

2. Modify source code and copy and distribute copies of modified source code

3. Copy and distribute copies of object code

License Limitations:

Modified source code

Mark that source code has been changed (change log)

All source code

Mark with copyright notice

Mark with disclaimer of warranty

Keep intact with all other notices

Object code

Accompany with source code

Written offer to provide source code upon request

License Obligations:

Must include a copy of the license in documentation available to the end-user

Must inform user of how to obtain the source code and that it is covered by GPL v2

Must redistribute source code [including modifications, if any]

When redistributing source code, must include a copy of the license

Source code modifications must be clearly marked as such and carry a

図 27：GPL v2 ライセンス プレーブック例（説明目的のみに使用）

ライセンス両立性マトリックス

ライセンス両立性とは、ソフトウェア コンポーネントとそのライセンスが一つ以上の他のコンポーネントとそのライセンスと両立できるかどうか（ライセンスが矛盾を起こさないか）を決めることです。両立性は、また、二つやそれ以上のライセンス（結合されたアウトライセンス）を結合した作成物に対する適切なライセンスについても言及します。

ライセンス両立性に関する取り組みは、両立しない条件の下に頒布されているさまざまなオープン ソース ソフトウェア コンポーネントを、ソースやオブジェクト形式で結合する際に、生じます。そのような結合は、純粋に法的な理由から再頒布できないようなソフトウェア コンポーネントの集合、つまり、ライセンス キメラを作り出します。

両立しないライセンスの例は、Apache version 2 ライセンスで頒布されたコードを GNU GPL version 2 の（古い GPL ライセンスには存在していなかった特許終了と免責に関する事項によって）ソフトウェアと結合しようとするときに見られます。ライセンス両立性の例は、明らかに GPL version 2 と両立できる、X11 ライセンスのコードを結合する時に見られます。

図 28 は、異なるライセンスで許諾された複数のソースからなる一つのソース コンポーネントを生成する例を示します。このシナリオでは、ソースは矛盾を抱えることなしにバイナリやオブジェクトファイルに結合できるような両立性のあるライセンス条件を持つことを確認しなければなりません。

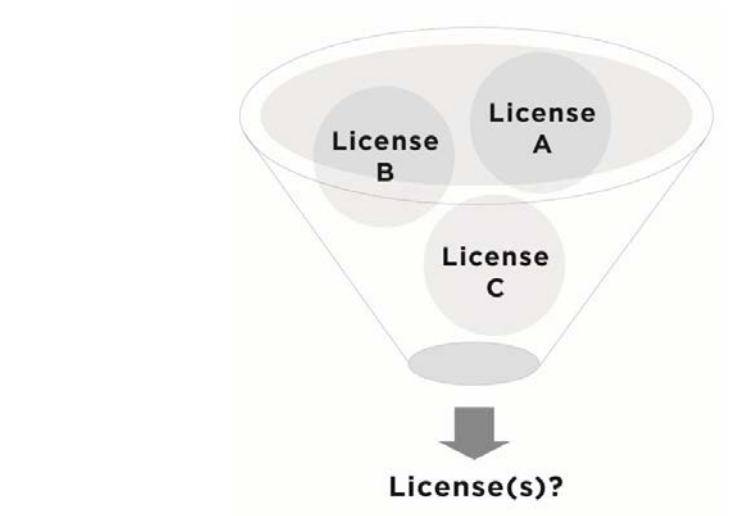


図 28：異なるライセンスで許諾されたソースを一つのバイナリへ結合する

ライセンス両立性は、開発チームが法務担当からの詳細なガイドラインを必要とするもので、自分たちだけで結論を出すべきではない分野です。そのようなガイドラインは、一般的なほとんどのライセンスをカバーするライセンス両立性マトリックスによって得られます。マトリックスの例を、表 7 に示します。

表 7：ライセンス両立性マトリックスの例（説明目的のみに使用）

	License-A	License-B	License-C	License-D	License-E	License-F	License-G
License-A	✔				✔	✔	
License-B		✔					
License-C			✔				
License-D		✔		✔			✔
License-E					✔		
License-F			✔				
License-G	✔						✔

開発チームは、異なるライセンスで許諾されたコードを結合する必要がある時には、問題となっている結合したソフトウェア コンポーネントがライセンス的な矛盾を発生させるかを確認するために、このマトリックスを参照できます。マトリックスに含まれていない新しいライセンスに出会った時には、そのライセンスは法務担当によって分析されなければなりません。法務担当は、分析結果に基づいて表の更新をするべきです。

ライセンス分類

法務担当が受ける質問数を減らし、ライセンスとコンプライアンス プロセス教育を増やすために、いくつかの会社では製品に使用する最もよく利用するライセンスをいくつかのカテゴリに分類しています。図 29（次ページ）は、最もよく使用するライセンスを 4 つのカテゴリに分類している、ライセンス分類の例です。

事前承認ライセンス

許容型オープン ソース ライセンスは、しばしばこのカテゴリに分類されます。これらのライセンスで許諾されたソース コードは、開発者が、マネージャや法務担当への承認プロセスを通すことなく、利用が事前承認されています。そのような事前承認は、開発者に、全ての通知を抜き取って、文書チームへ送付するように要求します。

マネージャ承認を必要とするライセンス

通知に関する義務（ライセンス文、帰属通知、著作権通知などの表記）に加えて、ソース コード変更を開示する義務があるので、これらのライセンスで許諾されたコンポーネントでは、マネージャによる承認が要求されます。

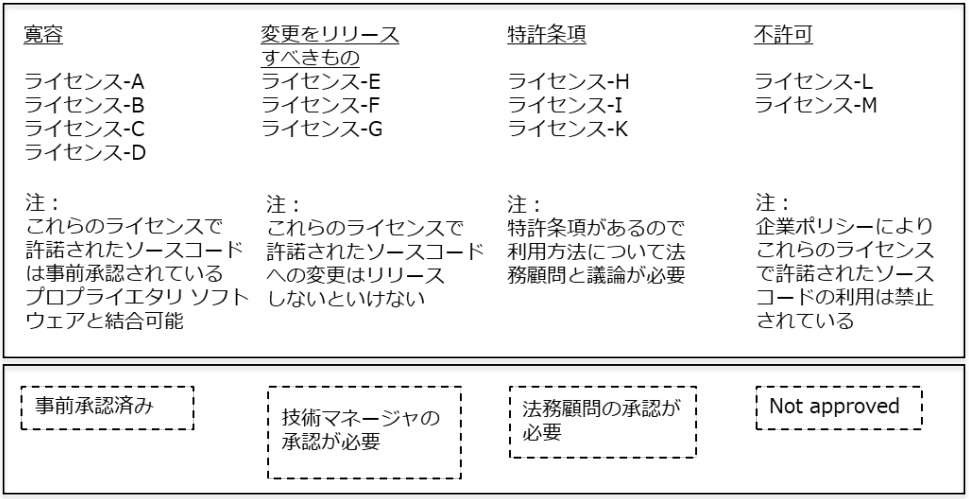


図 29 ライセンスカテゴリの例（説明目的にのみ使用）

法務担当による承認を必要とするライセンス

これらのライセンスで許諾されたソース コードは、法務によるレビューを承認を必要とします。これは、通常、特許に関する条項を含むライセンスに適用されます。

禁止されたライセンス

いくつかの企業では、特定のライセンスには「不許可」のフラグを付けます。企業ポリシーによって利用が許可されていません。

ライセンス分類はいかに役に立つか？

上記のライセンス カテゴリは、これらのライセンスで許諾されたコードを統合する際に、開発者が適切な一連の行動を簡単に知ることができるように分類する一つの方法です。さらに、ライセンスとなすべき行動との関連付けを簡単に作成することができます。表 8 は、開発者がさまざまなライセンスに関連した適切な行動を思い出すのに簡単な一つの方法です。

表 8：ライセンス分類のための簡単なハウツー

どのライセンス	行動
License A	問題なく利用
License E	マネージャの承認をもらう
License I	法務に相談する
License M	このソース コードは使えない
Other	一連の行動についてはマネージャに質問する

これらの異なるシナリオは、説明目的のみに提示されています。自分の組織ポリシーやガイドラインに従って、異なる行動を割り当てるような異なる分類モデルを作成することも可能です。

ソフトウェア相互作用法

コンプライアンス プロセスの一部として、通常、アーキテクチャー レビューがあります。その目的は、得的のソフトウェア コンポーネントが他のソフトウェア コンポーネントとどのように相互作用するかと、相互作用の方法を理解することです。アーキテクチャー レビューは、以下を特定します。

- （そのまま、または変更して利用される）オープン ソースからなるコンポーネント
- プロプライエタリ コンポーネント

- サード パーティ プロバイダーから提供されたものに由来するコンポーネント
- コンポーネントの依存状態
- 共有ヘッダ ファイルの利用
- コンポーネントの実行時のコンテキスト(Kernel/dirver/modules、middleware、libraries、applications、etc.)
- API を超えたコンポーネント間依存関係(s/w buses, IPCs, web APIs, etc.).
- 言語間の結合

表 9 と 10（次ページ）は、法務担当がソフトウェア開発者に提供できる追加情報を示します。この表は、どのライセンスが、企業ポリシーを考慮した上で、他のライセンスと動的静的にリンク可能であるかを示します。

表 9：動的リンク マトリックス例

動的リンクできるか	License-A	License-B	License-C	License-D
License-A	✔	✔	✔	✔
License-B		✔		✔
License-C	✔		✔	
License-D		✔	「事前承認必要」	✔

例えば、表 9 で、ライセンス B で許諾されたソース コードは、ライセンス D で許諾されたソース コードと動的にリンクすることが可能です。しかしながら、ライセンス C で許諾されたソース コードは、ライセンス B で許諾されたソース コードと動的にリンクすることはできません。また、リンクは、ライセンス間で常に相互に成り立つとは限らないに注意して下さい。

同様に、表 10 で、ライセンス A で許諾されたソース コードは、ライセンス C で許諾されたソース コードに静的にリンク可能です。しかしながら、ライセンス A で許諾されたソース コードは、ライセンス B で許諾されたソース コードに静的にリンクすることはできません。いくつかのリンクの組み合わせは、ケース バイ ケースで許可されます。そのため、「事前承認必要」と注意が書かれています。

表 10：静的リンク マトリックスの例

静的リンクできるか	License-A	License-B	License-C	License-D
License-A	✔		✔	
License-B		✔	「事前承認必要」	
License-C	✔		✔	
License-D	「事前承認必要」			✔

アーキテクチャー レビューによってリンクに関する課題（リンク マトリックスで示される企業ポリシーに従わない動的リンク静的リンクなど）が発見されたとき、アーキテクチャー レビューを推進した責任者（通常はコンプライアンス オフィサー）は、ソフトウェア コンポーネントに責任を持つソフトウェア開発者に通知して、改善を要求します。

チェックリスト

ほとんどの企業は、開発プロセスの全てのメジャーなマイルストーンで利用するチェックリストを確立しています。オープン ソース コンプライアンスで利用する時には、いくつかのチェックリストを作り上げ、製品のソース コード リポジトリへ外部の新しいオープン ソース コードを入れる前にチェックリストを使います。一つの例が、外部向けウェブサイトにソース コードを置く前に利用される、以下のチェックリストです。

- 全てのソース コード コンポーネントは、対応するコンプライアンス チケットを持っています。
- 全てのコンプライアンス チケットは、技術担当と法務担当に承認されています。
- 全てのコンプライアンス チケットは、付属している未解決サブタスクがクリアされています。

- 全てのソフトウェア コンポーネントの通知は、文書チームへ送付され、製品文書に含まれています。
- 法務担当は、書面による申出とコンプライアンス文書全体を承認しています。
- ソース コード パッケージは、準備され、標準開発マシン上でコンパイルできるかテストされます。
- 提供されるソース コードは、完結しており、製品に搭載されるバイナリと一致します。

そのようなチェックリストは、誤る確率を最小化し、オープン ソース管理に携わる全ての人がプロセスの次の段階へ進む前に何をすべきかを気づくようにさせます。

結論

ソフトウェア開発者は、統合し利用するさまざまなオープン ソース コンポーネントのライセンスに関しての教育を受ける必要があります。法務担当が実際的な方法で教育を実施するようにすることは、とても役に立ちます。ソフトウェア開発者が、法務に関連する日々の質問に答えてくれる文書化された実際的なアドバイスに、アクセスできるからです。この実際的なアドバイスは、通常以下の点を中心に展開します。

- オープン ソース コンポーネントをプロプライエタリやサード パーティ ソース コードへの挿入、またはその逆。
- オープン ソース コンポーネントをプロプライエタリやサード パーティ ソース コードにリンク、またはその逆。
- さまざまなソフトウェア コンポーネント間（プロプライエタリ、サード パーティ、オープン ソース）の相互作用法
- オープン ソース コンポーネントを利用する際に満足しなければならないライセンス義務

オープン ソース コンプライアンスは、コンプライアンス プログラムを確立し、コンプライアンス ポリシーとプロセスを作成し、実行する担当者を確立し、コンプライアンスの自動化の面で助けとなるツールをチームが使えるようにすれば、容易に達成できます。

著者について

イブラヒム ハダッド（Ph.D.） は、Samsung Electronics Co. Ltd.（韓国）の R&D に関する 100%子会社である Samsung Research America の R&D 担当副社長で、オープン ソース グループ長です。彼は、Samsung のオープン ソース戦略立案と実行、社内外との共同 R&D プロジェクト、重要オープン ソース開発プロジェクトへの参画に責任を持っており、さまざまなオープン ソース財団とオープン標準組織で Samsung の代表を務めています。

Samsung に加わる前、ハダッドは Linux Foundation の経営チームにおいて技術と法務コンプライアンス プロジェクトとイニシアチブの責任者でした。ハダッドは、キャリアを Ericsson Research で始めました。そこで彼は、5 年間ワイヤレス IP ネットワークに関する先端研究と、通信業品質環境への Linux とオープン ソース ソフトウェアの導入に携わりました。その後、彼は Motorola にオープン ソース技術グループの技術担当役員として加わり、Motorola のオープン ソースでのイニシアチブに貢献しました。

Motorola を離れた後、彼は、Palm において、webOS オープン ソース戦略とコンプライアンスに責任を持つオープン ソース担当役員としてオープン ソース部門を指揮しました。後に彼は、webOS をオープン ソース化した open webOS プロジェクトで、コンサルティングとして、Hewlett Packard を支援しました。

ハダッドは、優秀な成績で計算機科学の博士号を Concordia University (Montreal, Canada)から授与されました。彼は、学士と修士（ともに計算機科学）を Lebanese American University で取得しています。彼は、Linux Journal の寄稿編集者の一人で、Red Hat Linux と Fedora に関する 2 つの本の共著者であり、Linux System Administration、Fedora Linux、Ubuntu Linux に関する 4 つの本の技術監修者です。彼は、オープン ソース法務コンプライアンスから、ビジネス戦略や共同開発やイノベーションを推進する R&D ツールとしてのオープン ソース利用に至る話題についての著書と講演で知られています。

ハダッドは、アラビア語、英語、フランス語が堪能です。

Twitter: @IbrahimAtLinux

THE LINUX FOUNDATION

The Linux Foundation は、Linux の普及促進、保護、ならびに標準化に取り組み、Linux/OSS がクローズドなプラットフォームに対抗するのに必要とされる統合されたリソースとサービスを提供します。

The Linux Foundation およびその他の活動については、http://www.linuxfoundation.org を参照してください。