freeeにおけるOAuth/OIDCの活用とAuthleteへの移行

2024.12.11

IAM基盤開発部エンジニア

寺原 歩（てらら）

terahara ayumu

金融系SIer、SESを経て2021年にフリー株式会社（以降、freee）に入社。以降、IAM基盤開発部にてID連携開発に従事。

期せずして Authlete人材 としてfreeeが3社目。

freeeではエンジニアと兼務でDeveloper Relationshipを担当し、freee Tech Nightやfreee技術の日テックカンファレンスの運営などを行っていた。

趣味はホラー映画と猫と兎田ぺこら。

本日弊社から参加している愉快なメンバーたち

OAuth & OpenID Connect 勉強会 ー 認可サーバーの作りかた（AWS編）

2024/10/30 OAuth & OpenID Connect 勉強会 ー 認可サーバーの作りかた（AWS編）のYouTubeコメント欄より
https://authlete.connpass.com/event/333513/

目次

目次

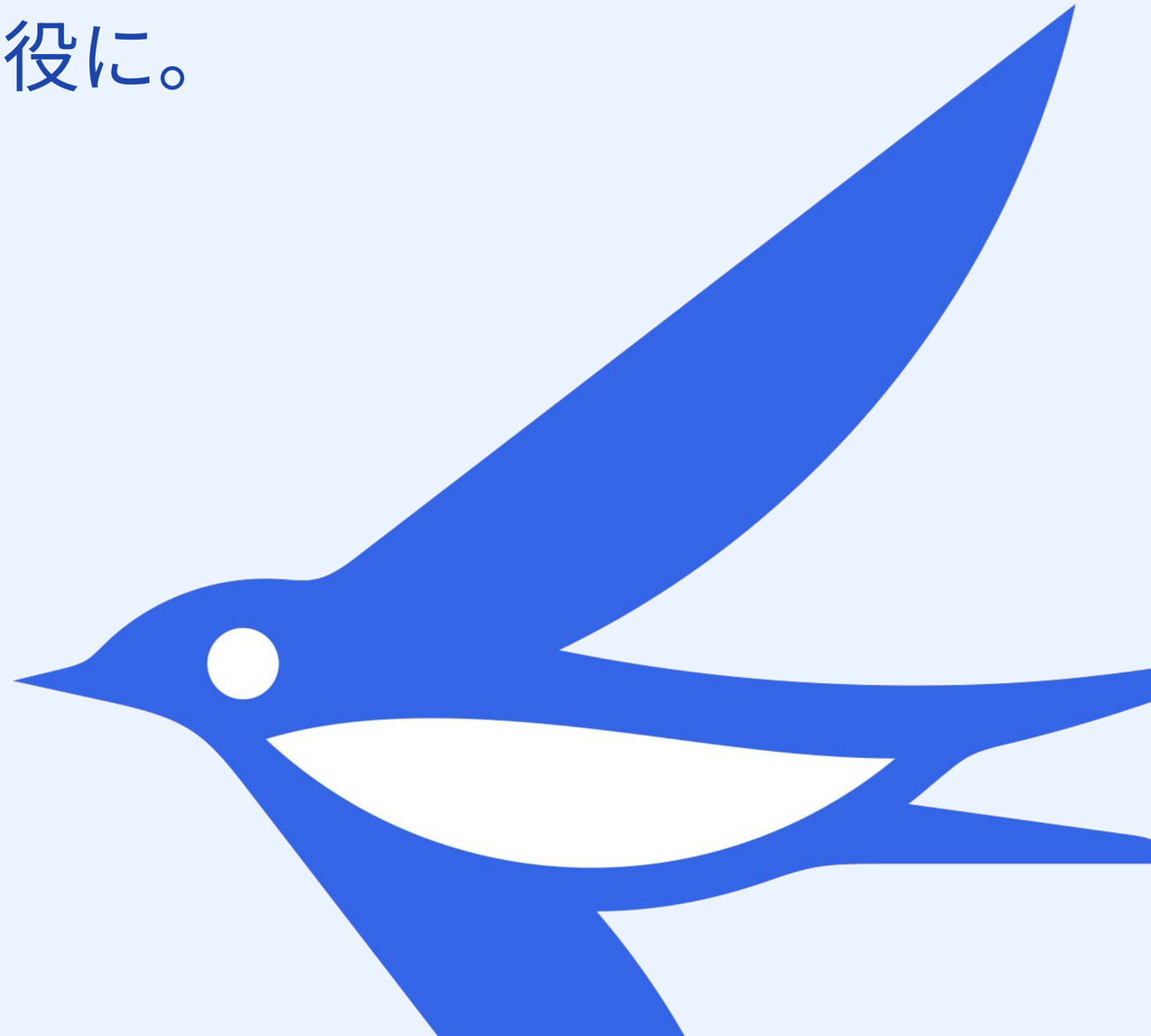
Mission

スモールビジネスを、世界の主役に。

freeeは「スモールビジネスを、世界の主役に。」をミッションに掲げ、
統合型経営プラットフォームを開発・提供し、
だれもが自由に自然体で経営できる環境をつくっていきます。

起業やビジネスを育てていくことを、もっと魅力的で気軽な行為に。
個人事業や中小企業などのスモールビジネスに携わるすべての人が、
じぶんらしく自信をもって経営できるように。

大胆にスピード感をもってアイデアを具現化できるスモールビジネスは、
今までにない多様な価値観や生き方、
新しいイノベーションを生み出す起爆剤だと私たちは考えています。
スモールビジネスが大企業を刺激し、社会をさらにオモシロク、
世の中全体をより良くする流れを後押ししていきます。



会計・人事労務・販売管理を核とした

統合型経営プラットフォーム

パブリックAPIによる拡張性/ freeeアプリストア

アプリストア掲載数(1)

179件

注1:2024年8月末時点

freeee アプリストア

～ freeeだけではできないことをアプリを使えば簡単に ～

- 様々な3rd partyアプリを簡易的に開発いただき、公開することができる
- ユーザーの選択肢の幅が広がるエコシステムの構築

freee モバイルアプリ

～ いつでも・どこでも利用できる機能の提供 ～

- freee人事労務モバイル
  - 位置情報による自動打刻、ウィジェットからの打刻などより利便性に特化した機能提供
- freee経費精算モバイル
  - 証憑を撮影して申請する「魔法スキャン」機能や「複数枚同時撮影」機能の提供
  - 交通系ICカード読み取りなど多彩な申請機能の提供
- freee電子申告モバイル
  - マイナンバーカード読み取りを使った電子申告の簡略化
- などなど

グループプロダクトへのfreeeアカウントログイン

- OpenID Provider として グループジョインプロダクトのスムーズなアカウント統合
- ユーザー体験の一貫性確保

目次

急成長するfreeeプラットフォームにおける認可基盤の課題

事業成長による要件の拡大

1. アプリストア
   - 100以上の連携アプリ
   - 非公開アプリも存在
   - 各アプリごとの認可管理
   - APIアクセス制御の複雑化

2. freeeモバイルアプリ群
   - 人事労務 / 経費精算 / 電子申告 / etc
   - アプリ固有の認証要件

3. グループプロダクト統合
   - OpenID Providerとしての役割
   - 統合的アカウント体験
   - 権限管理の複雑化

現行の認可基盤の限界

1. 技術的課題
   - RFC準拠の不完全性
   - 新規格対応の困難さ
   - セキュリティ更新の負担

2. 運用面の課題
   - 増大する保守コスト
   - 複雑化する監視要件
   - 属人化するナレッジ

3. スケーラビリティの課題
   - トラフィック増加への対応
   - 新規サービス追加時の負担
   - パフォーマンスボトルネック

急成長するfreeeプラットフォームにおける認可基盤の課題

解決が必要な重要課題

- ✓エコシステムの持続的な成長
- ✓セキュリティ品質の確保
- ✓運用効率の最適化
- ✓開発リソースの適切な配分

認可基盤の刷新

基盤移行の要件

現行基盤の状態

- 現行はフルスクラッチによる実装
- 旧認証認可基盤から認証基盤のみを切り離した実績があり、現行基盤は認可機能のみ残存

移行先必須要件

- 既存の認証基盤との親和性
- freee独自の事業所アカウントとの親和性

移行コストの比較要件

- インフラにかかる初期コストおよびランニングコスト（Authlete利用費含む）
- ミドルウェアアップデートにかかる運用コスト
- セキュリティアップデート拡張仕様への対応コスト（PKCE、FAPI2.0など）
- その他拡張仕様への対応コスト（CIBAなど）

移行先の選定

	フルスクラッチ	Authlete	IDaaS A
既存の認証機能との親和性	●	●	
事業所アカウントとの親和性	●	●	
費用(月額)	●		
運用コスト		●	
拡張仕様開発コスト		●	
その他	ベンダーロックイン無 ID人材不足	エンハンスに強い 設計からサポート有	必須要件未達のため対象外

Authlete採用理由

事業成長に耐えうる基盤構築が可能

- 事業成長による要件の拡大に耐えうる基盤を作るためにはOAuthの拡張機能の開発がネックであるが、Authleteを採用することにより開発が非常に軽くなることが見込まれる。
- Authleteを採用することで認可基盤の設計/開発サポートも受けられるので、今後の開発においてそれら実装周りで困ることはなく、今後のセキュリティアップデートも素早く追従できる。
- 競合他社では5年以上の運用実績があり、他社SaaSも導入していることから信頼性も高い。競合他社とのセキュリティ、拡張仕様面でのビジネス機会を失うことがない。

コスト要件を満たす

- インフラ月額費用に関しては対フルスクラッチ自社サーバー費用ではAuthlete費用の方が高い。
- 一方、その差額はフルスクラッチのためにOAuthに熟達したエンジニアを一人雇うよりは安いと考えている。それら運用コスト / 体制維持へのコストまで含めるとフルスクラッチよりもAuthleteを採用した方が安価。

freee 新認可基盤

x

Authlete

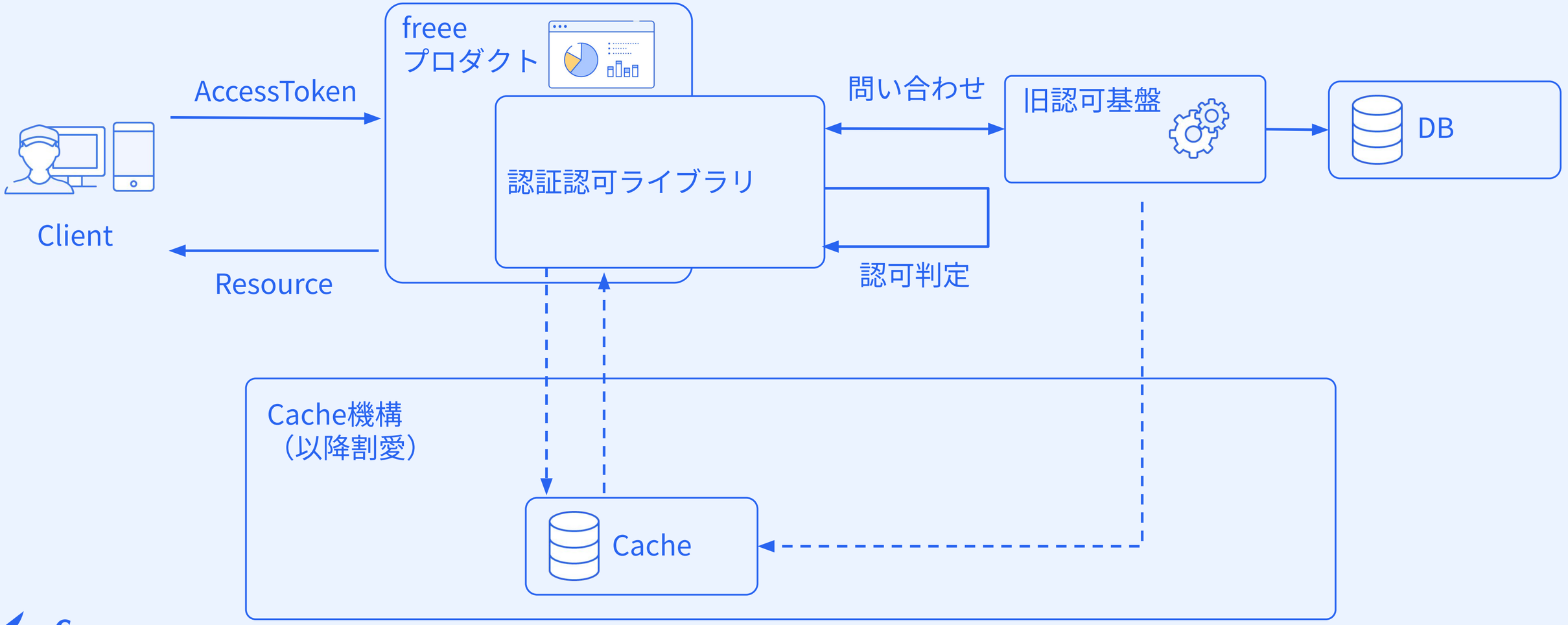
目次

freeeの技術スタック

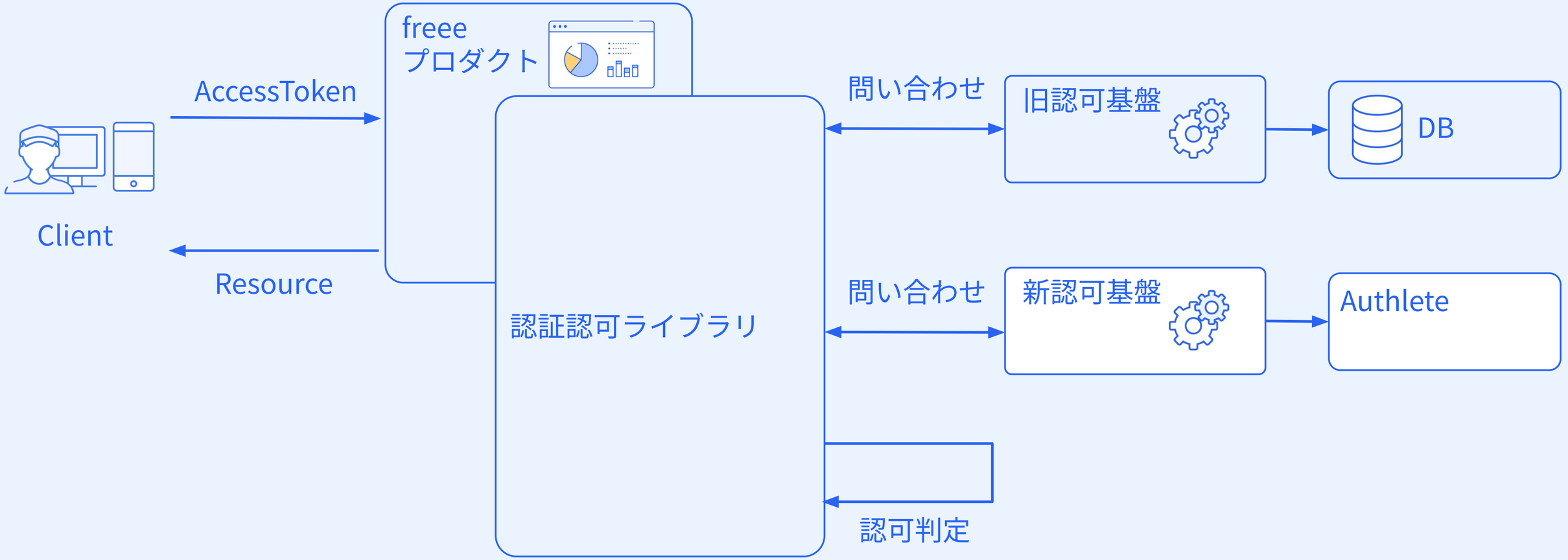
全領域で技術入れ替えがほぼ一巡。

レイヤ	2012 - 2013		現在	
バックエンド	Ruby	Rails	Ruby Go	Rails
Webフロントエンド	CoffeeScript	Backbone.js Sprockets	TypeScript JavaScript (Flow)	React Vite / webpack
モバイルアプリ	-	-	Swift / Kotlin	SwiftUI / Jetpack Compose
インフラ・構成管理	Heroku -> AWS	Chef	AWS	Kubernetes Terraform
ML・データ分析	-	-	Python	TensorFlow BigQuery

旧環境のシーケンス（例：Public APIリクエスト）



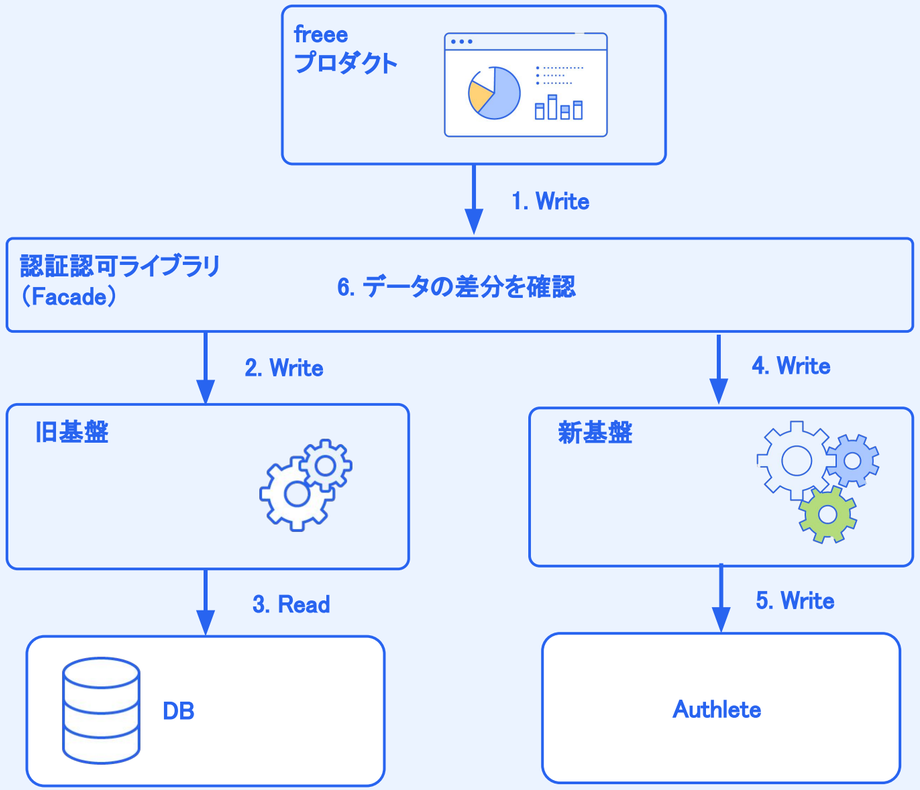
新環境へ移行するシーケンス（例：Public APIリクエスト）



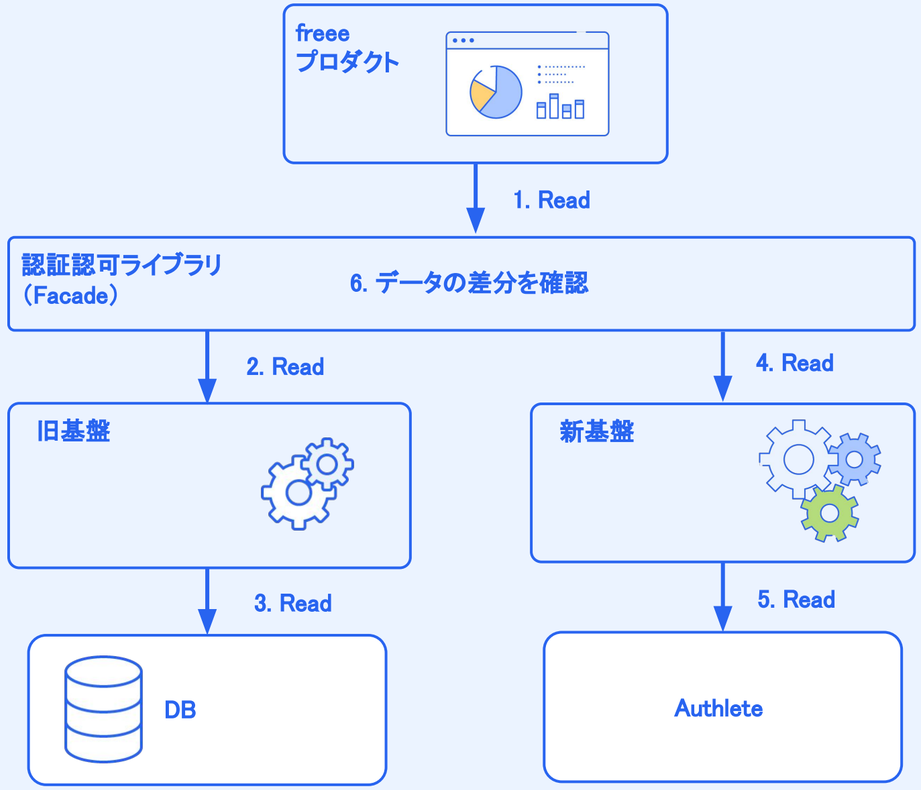
ストラングラーフィグパターン

- 認証認可ライブラリをストラングラーファサードとする
- 移行戦略として、二重書き込みにてデータを揃え、二重読み込みにてデータ差分・機能差分を検知する
- プロダクト側は意識せず、新基盤に移行可能

二重書き込み



二重読み込み



移行フェーズの切り方

- Phase1: アプリケーション情報の移行
  - freeeアプリストアから作成されるデータを中心にまずはOAuth Clientの移行を行う
  - アクセストークンとは異なりfreee独自ドメインを持つ機能が多く、再モデリングが必要
  - レガシーコードも大量にあるため技術的負債の返済が主軸
- Phase2: アクセストークン情報の移行
  - 独自ドメインを持つ機能はないため、機能面は単純な移行が可能
  - アクセス量、データ量が多く、パフォーマンスチューニングが必要
  - Invalidなアクセストークンリクエストが大量にあるため、Client開発者との折衝が主軸

各Phaseごとに、さらにストラングラーフィグを満たすための各タスクを順次実行していく

- Secondary Write: 二重書き込みにより新基盤へデータを順次流し込む
- Secondary Read: 二重読み込みにより新基盤のデータ・I/Fが現新一致することを保証する
- Primary Read: 読み込み機能の主系・副系を入れ替えて新基盤のみで動作可能であることを保証する
- Primary Write: 書き込み機能の主系・副系を入れ替えて新基盤のみで動作可能であることを保証する

移行フェーズの切り方まとめ

	Ph1	Ph2
Secondary Write	① OAuth Client情報の順次移行	⑤ AccessToken情報の順次移行
Secondary Read	② OAuth Client情報の差分修正	⑥ AccessToken情報の差分修正
Primary Read	③ OAuth Client機能の主副切り替え	⑦ AccessToken機能の主副切り替え
Primary Write	④ OAuth Client機能の主副切り替え	⑧ AccessToken機能の主副切り替え
補足	技術的負債の返済フェーズ	パフォーマンスチューニングフェーズ

ストラングラーフィグとAuthleteの相性の良さ

freeeの仕様をブラッシュアップ

- Authleteの特性として、RFC準拠の質と速さの両方を備えている
- ストラングラーフィグにて機能差分が出た時に一般的には主系に合わせた改修を行う必要があるが、副系であるAuthleteを使った新基盤の方が機能面で優秀なケースが多かった
- 副系に合わせてfreeeの仕様を更新することで今まで見えていなかったリスクを回避
- 実際の改善例: Relying Partyがトークンリフレッシュ時にid_tokenを再発行していたが、全てのRelying Partyはこれを利用していなかったので削除した

メンバーのスキルアップ

- Authlete API I/F と 各種RFC を開発チームで読み合わせしながら進めることが習慣付き、設計品質も向上した
- 川崎さんの解説記事をチーム内共有する時は :taka_god: というemojiリアクションが付くように

目次

Authleteのオプション一覧

移行に利用したオプション

- ClientID Alias
  - 旧基盤のClientIDをそのまま利用可能
- Client Attributes
  - freeeのClientに関するドメイン情報の判定に使う値を設定することで機能の現新一致が可能
- /auth/token/create API
  - 旧基盤で発行した AccessToken をそのまま利用可能
- AccessToken properties
  - freee独自の事業所アカウントなどのドメイン情報を設定することで機能の現新一致が可能

移行後に使いたいオプション

- 冪等性リフレッシュトークン

AccessTokenキャッシュの実装方法

前提

- AuthleteではAccessTokenをハッシュ化して保持している（例: List Issued Tokens ）
- freeeではAccessToken/RefreshTokenを指定するRevocation以外に、Resource OwnerやOAuth Clientを指定して一括でRevocationを行う機能がある

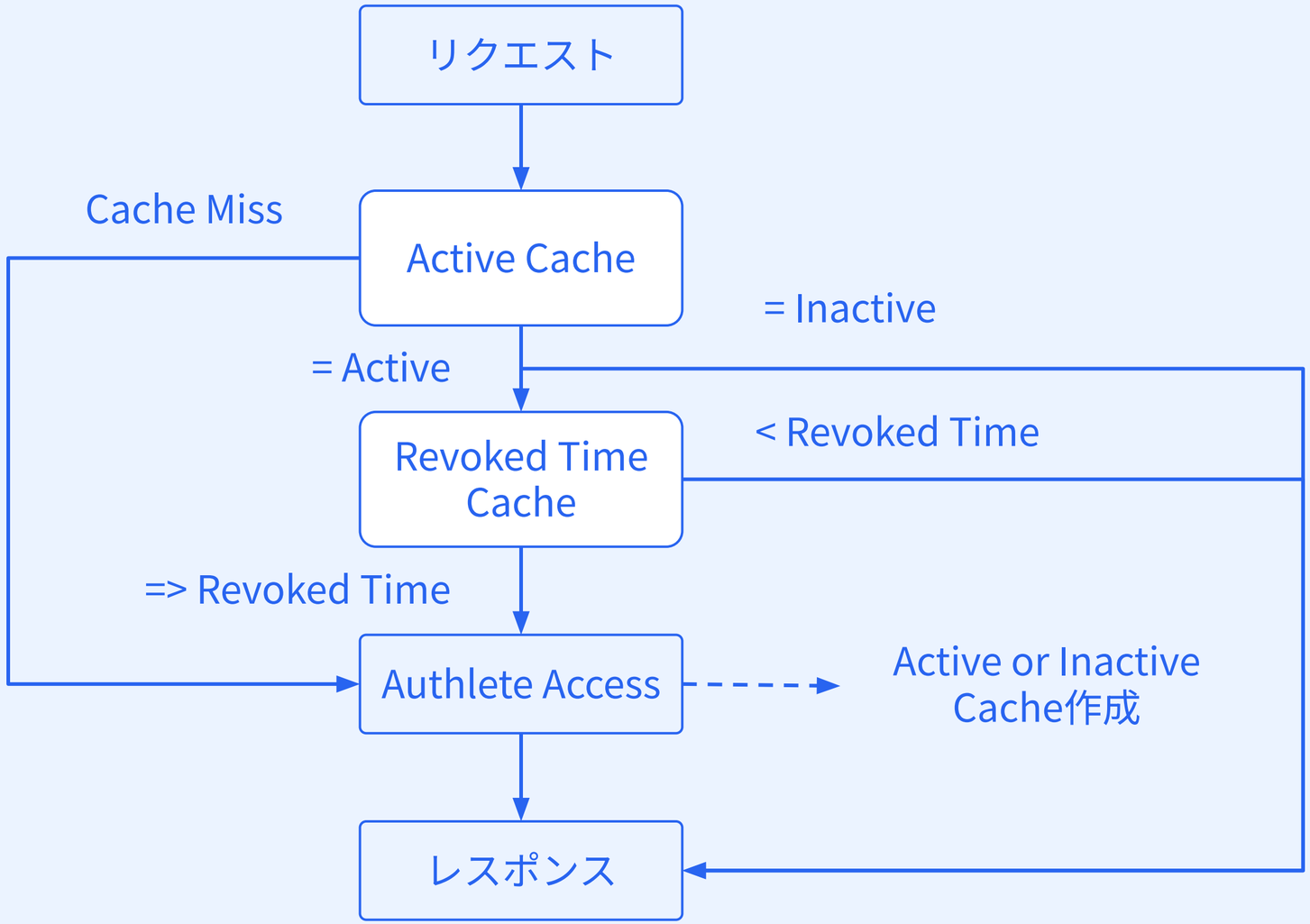
キャッシュ要件

- Token IntrospectionによるAuthleteへのリクエスト負荷を下げる
- 一括でToken Revocationされた場合にAccessTokenキャッシュを無効化できる

キャッシュ種別

- Active AccessToken キャッシュ: Token Introspectionで有効なキャッシュ情報を持つ
- Inactive AccessToken キャッシュ: Token Introspectionで無効なキャッシュ情報を持つ
- Revoked Time キャッシュ: 一括でRevocationを行った際のResource OwnerとOAuth Clientをkeyに持ち、Revoked Time と Active AccessToken キャッシュ の作成日時を比較し、有効 / 無効を判定する

AccessTokenキャッシュの実例（Token Introspection）



自社特有のドメインの実装方法について

自社特有のドメインをパターン化することで設計の見通しが格段に向上
さらに Authlete では Client Attributes にてパターンに使う識別子を管理することが可能
という「OAuth/OIDCのClient管理を"型"で制する - freee_client_typeの軌跡」という記事を書いたのでぜひB!ブックマークお願いします。

目次

OAuth Client管理機能の拡充

- OAuth Clientを一括操作するAPIがないため、アプリストアを提供するための機能に物足りなさ
- Client Attributesを指定した検索機能など、List系の機能があると嬉しい

authlete-go のメンテナンス終了が悲しい

- 新基盤はgolangを使っているため、 authlete-go を利用していたところ新規commitが…！

We strongly recommend the use of https://github.com/authlete/openapi-for-go instead of authlete-go 😇

- authlete-go から openapi-for-go へ移行する場合に型定義が破壊的変更になるため、移行が容易ではない
- openapi-for-go の生成コードをそのまま利用するよりもOpenAPI定義ファイルからcodegenする方が生成コードの自由度が高い
- authlete-goを弊社と共にcontributeしていきませんか…？

管理コンソールのアクセス制御機能の強化

- Authlete 3.0リリースおめでとうございます！お待ちしておりました！

目次

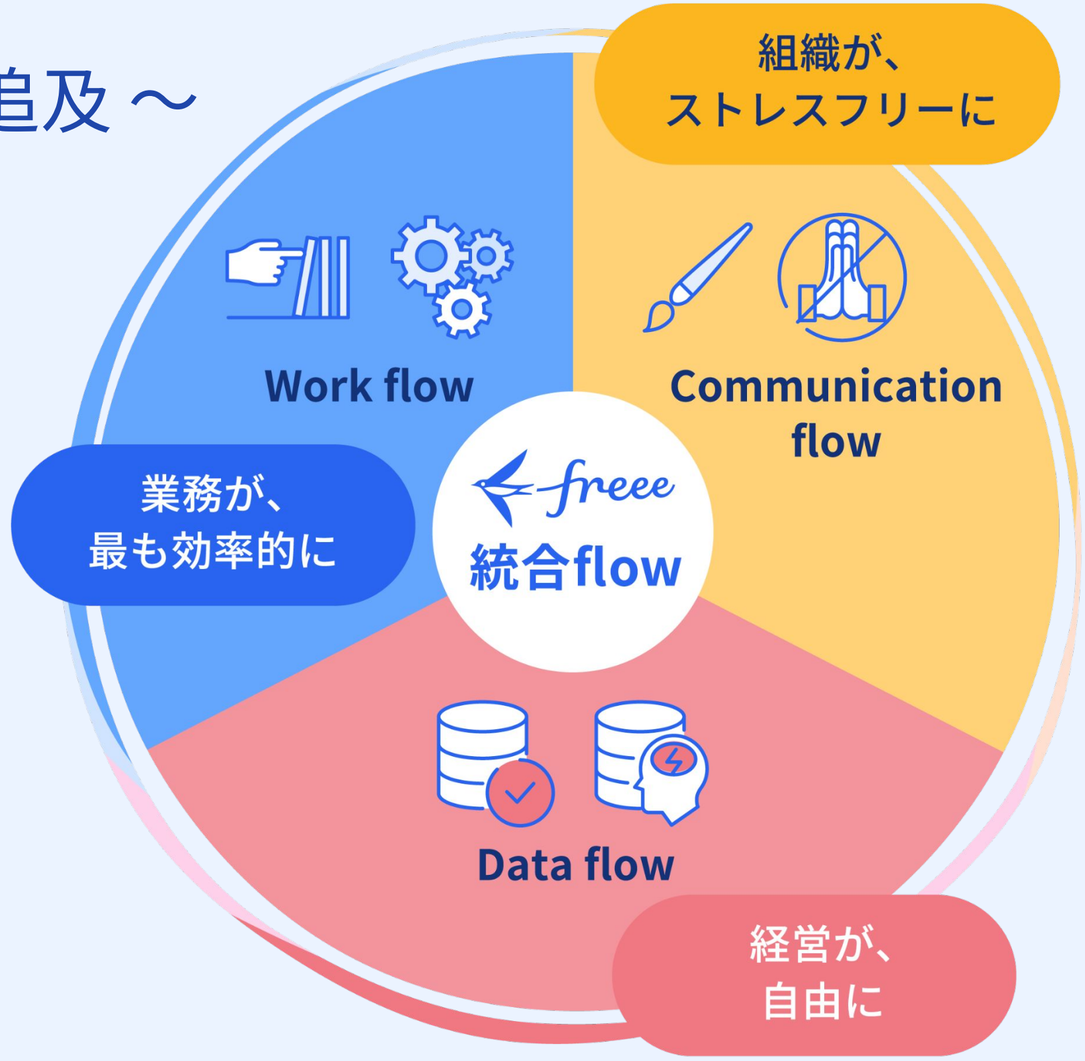
プロダクトのビジョン

～ 統合型経営プラットフォームの追及 ～

- 業務が、最も効率的に
- 組織が、ストレスフリーに
- 経営が、自由に

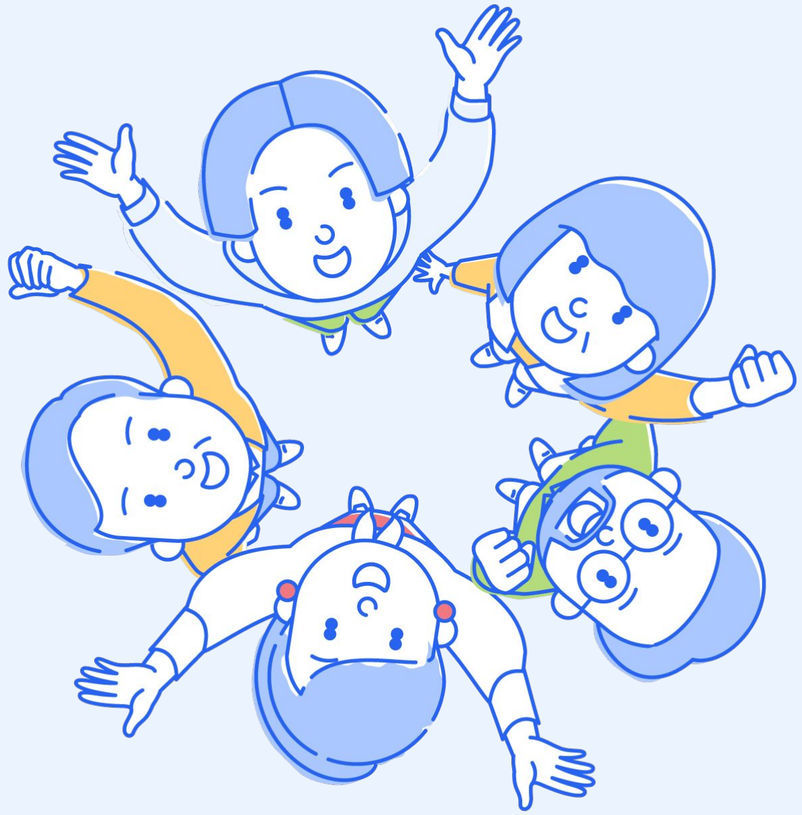
経営を阻む３つの分断を解決する 統合flow

スモールビジネスに最高の体験を最速で提供するために。
進化可能なアーキテクチャを。



toB SaaS 認可のベストプラクティスへ

- Resource Ownerが複雑化するtoB SaaSで正しいResource Ownerへ認可を取ることは難易度が高い
  - 事業所情報のOwnerは誰なのか。
- 認可機能に特化しているAuthleteと共にtoB SaaS認可のプラクティスを追及していく



スモールビジネスを、世界の主役に。