

OpenID Connect と SCIM の エンタープライズ実装ガイドライン



一般社団法人 OpenID ファウンデーション・ジャパン
Enterprise Identity ワーキンググループ (EIWG)
技術タスクフォース 著

作成日：2016年3月28日

リビジョン：1.0

目次

| | | |
|------------|--|----|
| 第 1 章 | はじめに | 9 |
| 1.1. | 本実装ガイドの構成 | 9 |
| 1.2. | 本実装ガイドが参照している技術標準 | 10 |
| 第 2 章 | 標準プロトコル超入門 | 11 |
| 2.1. | フェデレーション向けプロトコル OpenID Connect | 11 |
| 2.1.1. | OpenID Connect とは | 11 |
| 2.1.2. | OpenID Connect を使った認証フロー | 11 |
| 2.1.2.1. | 認可コードフロー | 12 |
| 2.1.2.2. | Implicit フロー | 13 |
| 2.1.3. | ID トークン | 14 |
| 2.1.4. | ユーザーの属性情報の受け渡し方法 | 15 |
| 2.1.4.1. | ユーザーの属性情報の要求 | 15 |
| 2.1.4.2. | ユーザーの属性情報の受け取り | 17 |
| 2.1.4.2.1. | ID トークンに含める方法 | 17 |
| 2.1.4.2.2. | UserInfo エンドポイントを通じて提供する方法 | 17 |
| 2.1.5. | セッション管理に関する関連仕様 | 18 |
| 2.2. | プロビジョニング向けプロトコル SCIM | 18 |
| 2.2.1. | SCIM とは | 18 |
| 2.2.2. | SCIM で扱えるデータ | 19 |
| 2.2.2.1. | SCIM で扱える属性の型 | 19 |
| 2.2.2.2. | ユーザーリソーススキーマ | 21 |
| 2.2.2.3. | Enterprise User スキーマ拡張 | 23 |
| 2.2.2.4. | スキーマ定義のためのスキーマ | 23 |
| 2.2.3. | SCIM を使ったデータ操作 | 24 |
| 2.2.3.1. | HTTP メソッドと操作 | 25 |
| 2.2.3.2. | リソース (データ) に対する操作 | 25 |
| 2.2.3.3. | エンドポイントアクセス時の認証方法 | 25 |
| 2.2.4. | SCIM 1.1 から SCIM 2.0 への変更点の要点 | 26 |
| 第 3 章 | モデルユースケースと実装の概要 | 28 |
| 3.1. | 利用企業の認証システムとクラウドサービスの連携モデル | 28 |
| 3.2. | 連携のための初期設定 | 29 |
| 3.2.1. | クラウドサービスが設定に必要な基本情報を提供する | 30 |
| 3.2.2. | 利用企業の ID 管理サーバーで SCIM クライアントしての設定を行なう | 31 |

| | | |
|----------|---|----|
| 3.2.3. | 利用企業の認証サーバーへクラウドサービスを OpenID Connect RP として登録する | 31 |
| 3.2.4. | クラウドサービスに認証サーバーを OpenID Connect OP として登録する | 32 |
| 3.3. | ユーザー情報のプロビジョニング | 33 |
| 3.3.1. | ユーザー属性 | 34 |
| 3.3.1.1. | 組織情報、役職情報のプロビジョニングを実装する際の考慮点..... | 37 |
| 3.3.2. | 特別な意味を持つ属性..... | 37 |
| 3.3.2.1. | ID 管理サーバー上の識別子 (externalId) | 37 |
| 3.3.2.2. | サービス利用時のユーザー名 (userName) | 38 |
| 3.3.2.3. | 企業 OP でのログイン ID (externalUserName)..... | 38 |
| 3.3.2.4. | ID トークンの Issuer と Subject (idTokenClaims.issuer, idTokenClaims.subject) | 38 |
| 3.3.2.5. | 有効フラグ (active) | 38 |
| 3.3.3. | ユーザー情報の操作 | 39 |
| 3.3.3.1. | ユーザーの作成 | 39 |
| 3.3.3.2. | ユーザーの更新 | 39 |
| 3.3.3.3. | ユーザーの削除 | 39 |
| 3.4. | ユーザー認証 | 40 |
| 3.4.1. | ID トークン | 40 |
| 3.4.2. | 認証フロー | 40 |
| 3.4.2.1. | クラウドサービス起点のログイン | 41 |
| 3.4.2.2. | 利用企業起点のログイン..... | 41 |
| 3.4.2.3. | 再認証要求 | 42 |
| 3.4.3. | ユーザーの紐付け | 43 |
| 3.5. | 強制ログアウト | 43 |
| 3.5.1. | ログアウトトークン | 43 |
| 3.5.2. | 強制ログアウト処理 | 44 |
| 3.6. | 鍵・パスワードのメンテナンス | 44 |
| 3.6.1. | ID トークン署名用キーペアの更新..... | 44 |
| 3.6.2. | SCIM エンドポイントアクセス用パスワードの更新 | 45 |
| 3.6.3. | SCIM エンドポイントアクセス時の OAuth を使った認可への対応 | 45 |
| 第 4 章 | クラウドサービス事業者向け実装ガイド | 47 |
| 4.1. | OpenID Connect RP (クライアント) の実装..... | 47 |
| 4.1.1. | 認証フローの概要 | 47 |
| 4.1.2. | ログイン開始エンドポイントの実装..... | 47 |
| 4.1.3. | 利用企業の認証サーバーへの認証要求 | 49 |
| 4.1.3.1. | 通常の認証要求 | 49 |

| | | |
|------------|--------------------------------------|----|
| 4.1.3.2. | 再認証のための認証要求..... | 50 |
| 4.1.4. | 利用企業の認証サーバーからの認証結果の受け取り | 50 |
| 4.1.4.1. | 認証成功時の応答..... | 52 |
| 4.1.4.2. | 認証失敗時の応答..... | 52 |
| 4.1.5. | ID トークンの検証 | 53 |
| 4.1.5.1. | 通常の認証要求の場合の ID トークンの検証..... | 53 |
| 4.1.5.2. | 再認証要求の場合の ID トークンの検証 | 55 |
| 4.1.6. | 認証されたユーザーとクラウドサービスのユーザーとの紐付け | 56 |
| 4.1.7. | セッションの管理..... | 56 |
| 4.1.8. | 強制ログアウト処理 | 56 |
| 4.1.8.1. | 利用企業の認証サーバーからのログアウト要求の受け取り | 56 |
| 4.1.8.2. | ログアウトトークンの検証..... | 56 |
| 4.1.8.3. | ログアウト対象のユーザーとクラウドサービスのユーザーとの紐付け..... | 57 |
| 4.1.8.4. | ログアウト処理成功時の応答 | 57 |
| 4.1.8.5. | ログアウト処理失敗時の応答 | 58 |
| 4.1.9. | 利用企業の認証サーバーの公開鍵の取得 | 58 |
| 4.2. | SCIM サーバーの実装..... | 59 |
| 4.2.1. | User エンドポイントの実装..... | 59 |
| 4.2.1.1. | ユーザーの作成 (POST)..... | 60 |
| 4.2.1.1.1. | リクエストの受け取り | 60 |
| 4.2.1.1.2. | クラウドサービスのユーザーとして登録..... | 60 |
| 4.2.1.1.3. | 登録結果の応答..... | 62 |
| 4.2.1.2. | ユーザーの更新 (PUT)..... | 63 |
| 4.2.1.2.1. | リクエストの受け取り | 63 |
| 4.2.1.2.2. | クラウドサービスのユーザーを更新..... | 63 |
| 4.2.1.2.3. | 更新結果の応答..... | 65 |
| 4.2.1.3. | ユーザーの削除 (DELETE)..... | 66 |
| 4.2.1.3.1. | リクエストの受け取り | 66 |
| 4.2.1.3.2. | クラウドサービスのユーザーを削除..... | 66 |
| 4.2.1.3.3. | 削除結果の応答..... | 66 |
| 4.2.2. | 検索エンドポイントの実装 | 67 |
| 4.2.2.1. | ユーザーの検索 | 67 |
| 4.2.2.1.1. | リクエストの受け取り | 67 |
| 4.2.2.1.2. | リクエストの検証 | 68 |
| 4.2.2.1.3. | 検索結果の応答..... | 68 |
| 4.2.3. | ユーザーデータの検証..... | 69 |
| 4.2.4. | エンドポイントアクセス時の認証 | 70 |

| | | |
|------------|-------------------------------------|----|
| 4.3. | 利用企業の管理者向け機能の実装 | 71 |
| 4.3.1. | OpenID Connect OP（認証サーバー）登録機能 | 71 |
| 4.3.2. | SCIM クライアント（ID 管理サーバー）登録機能 | 72 |
| 4.3.3. | 管理者向け機能利用時の認証方式 | 73 |
| 第 5 章 | クラウドサービス利用企業向け実装ガイド | 74 |
| 5.1. | OpenID Connect OP（認証サーバー）の実装 | 74 |
| 5.1.1. | 認証フローの概要 | 74 |
| 5.1.2. | クラウドサービスから認証要求を受け取る | 74 |
| 5.1.2.1. | 通常認証要求 | 74 |
| 5.1.2.2. | 再認証のための認証要求 | 76 |
| 5.1.3. | ユーザーを認証する | 77 |
| 5.1.4. | ユーザーに認証連携の同意を得る | 77 |
| 5.1.5. | クラウドサービスへ認証結果を返す | 78 |
| 5.1.5.1. | 認証成功時の応答 | 78 |
| 5.1.5.2. | 認証失敗時の応答 | 78 |
| 5.1.6. | ID トークン | 79 |
| 5.1.6.1. | JOSE ヘッダ | 79 |
| 5.1.6.2. | ID トークンに格納するクレーム | 80 |
| 5.1.6.3. | ID トークンの署名 | 80 |
| 5.1.7. | ユーザーセッションを強制ログアウトする | 80 |
| 5.1.7.1. | ログアウトトークン | 80 |
| 5.1.7.2. | クラウドサービスへのログアウトの要求 | 81 |
| 5.1.7.3. | ログアウト要求を送るクラウドサービスの選択方法 | 81 |
| 5.1.8. | 公開鍵の公開（JWK Set エンドポイントの実装） | 82 |
| 5.2. | SCIM クライアントの実装 | 83 |
| 5.2.1. | ユーザー情報の操作 | 83 |
| 5.2.1.1. | ユーザーの作成 | 83 |
| 5.2.1.2. | ユーザーの更新 | 84 |
| 5.2.1.2.1. | ユーザーリソース識別子 (id) の取得 | 85 |
| 5.2.1.2.2. | ユーザー情報の更新 | 86 |
| 5.2.1.3. | ユーザーの削除 | 87 |
| 5.2.1.4. | エラー応答 | 88 |
| 5.2.2. | エンドポイントアクセス時の認証 | 89 |
| 参考文献 | | 90 |
| 付録 A. | OpenID Connect リクエスト・レスポンス例 | 91 |
| A.1. | OpenID Connect ID トークン | 91 |

| | | |
|----------|---|----|
| A.1.1. | 通常の認証要求で応答される ID トークン | 91 |
| A.1.1.1. | JOSE ヘッダ | 91 |
| A.1.1.2. | クレーム | 91 |
| A.1.1.3. | ID トークン | 91 |
| A.1.2. | 再認証要求で応答される ID トークン | 91 |
| A.1.2.1. | JOSE ヘッダ | 91 |
| A.1.2.2. | クレーム | 92 |
| A.1.2.3. | ID トークン | 92 |
| A.2. | OpenID Connect による認証 | 92 |
| A.2.1. | 認証リクエスト | 92 |
| A.2.2. | 認証成功レスポンス | 92 |
| A.2.3. | 認証エラーレスポンス | 93 |
| A.2.3.1. | 認証に失敗した | 93 |
| A.2.3.2. | 同意画面でユーザーが拒否した | 93 |
| A.2.3.3. | scope の値に誤りがある | 93 |
| A.2.3.4. | response_type の値に誤りがある | 93 |
| A.2.3.5. | リクエストパラメータに誤りがある | 93 |
| A.2.3.6. | サーバーサイドの処理でエラーが発生した | 94 |
| A.3. | OpenID Connect による再認証要求 | 94 |
| A.3.1. | 認証リクエスト | 94 |
| A.3.2. | 認証成功レスポンス | 94 |
| A.3.3. | 認証エラーレスポンス | 94 |
| A.3.3.1. | 認証に失敗した | 94 |
| A.3.3.2. | 同意画面でユーザーが拒否した | 95 |
| A.3.3.3. | scope の値に誤りがある | 95 |
| A.3.3.4. | response_type の値に誤りがある | 95 |
| A.3.3.5. | リクエストパラメータに誤りがある | 95 |
| A.3.3.6. | サーバーサイドの処理でエラーが発生した | 95 |
| A.4. | OpenID Connect Back-Channel Logout ログアウトトークン | 96 |
| A.4.1. | JOSE ヘッダ | 96 |
| A.4.2. | クレーム | 96 |
| A.4.3. | ログアウトトークン | 96 |
| A.5. | OpenID Connect Back-Channel Logout による強制ログアウト要求 | 96 |
| A.5.1. | 強制ログアウト要求 | 96 |
| A.5.2. | 強制ログアウト成功レスポンス | 97 |
| A.5.3. | 強制ログアウトエラーレスポンス | 97 |
| A.5.3.1. | リクエストにエラーがある | 97 |

| | | |
|----------|--------------------------|-----|
| A.5.3.2. | ログアウトトークンの検証に失敗した..... | 97 |
| A.5.3.3. | ユーザーとの紐付けに失敗した..... | 97 |
| A.5.3.4. | ログアウト処理が失敗した..... | 97 |
| A.5.3.5. | 関連サイトのログアウト処理に失敗した..... | 97 |
| A.6. | JWK Set..... | 98 |
| 付録 B. | SCIM リクエスト・レスポンス例..... | 99 |
| B.1. | SCIM EIWG 拡張リソース..... | 99 |
| B.1.1. | EIWG 拡張属性..... | 99 |
| B.1.2. | EIWG 拡張スキーマ..... | 100 |
| B.2. | SCIM ユーザーの作成..... | 105 |
| B.2.1. | ユーザー作成リクエスト..... | 105 |
| B.2.1.1. | 作成するユーザーの属性..... | 105 |
| B.2.1.2. | SCIM リクエスト..... | 108 |
| B.2.2. | ユーザー作成成功レスポンス..... | 110 |
| B.2.2.1. | サーバーが付与する属性..... | 110 |
| B.2.2.2. | SCIM レスポンス..... | 110 |
| B.2.3. | ユーザー作成エラーレスポンス..... | 112 |
| B.2.3.1. | リクエストメッセージに誤りがある..... | 113 |
| B.2.3.2. | ユーザー属性値に誤りがある..... | 113 |
| B.2.3.3. | ユーザー属性値が重複している..... | 113 |
| B.2.3.4. | ユーザー作成処理中にエラーが発生した..... | 113 |
| B.2.3.5. | 認証に失敗した..... | 114 |
| B.2.3.6. | リソースへのアクセス・操作権限がない..... | 114 |
| B.3. | SCIM ユーザーの検索..... | 114 |
| B.3.1. | ユーザー検索リクエスト..... | 114 |
| B.3.1.1. | ユーザー検索条件..... | 114 |
| B.3.1.2. | SCIM リクエスト..... | 114 |
| B.3.2. | ユーザー検索成功レスポンス..... | 115 |
| B.3.2.1. | SCIM レスポンス..... | 115 |
| B.3.3. | ユーザー検索エラーレスポンス..... | 115 |
| B.3.3.1. | ユーザーが見つからない..... | 116 |
| B.3.3.2. | リクエストメッセージに誤りがある..... | 116 |
| B.3.3.3. | リクエストのフィルター設定に誤りがある..... | 116 |
| B.3.3.4. | ユーザー検索処理中にエラーが発生した..... | 117 |
| B.3.3.5. | 認証に失敗した..... | 117 |
| B.3.3.6. | リソースへのアクセス・操作権限がない..... | 117 |
| B.4. | SCIM ユーザーの更新..... | 117 |

| | | |
|----------|---|-----|
| B.4.1. | ユーザーリソース識別子取得..... | 117 |
| B.4.2. | ユーザー更新リクエスト..... | 118 |
| B.4.2.1. | ユーザー更新条件..... | 118 |
| B.4.2.2. | SCIM リクエスト..... | 118 |
| B.4.3. | ユーザー更新成功レスポンス..... | 120 |
| B.4.3.1. | サーバが更新する属性..... | 120 |
| B.4.3.2. | SCIM レスポンス..... | 120 |
| B.4.4. | ユーザー更新エラーレスポンス..... | 123 |
| B.4.4.1. | ユーザーリソースが見つからない..... | 123 |
| B.4.4.2. | リクエストメッセージに誤りがある..... | 123 |
| B.4.4.3. | ユーザー属性値に誤りがある..... | 124 |
| B.4.4.4. | ユーザー属性値が重複している..... | 124 |
| B.4.4.5. | ユーザーの更新要求が競合した..... | 124 |
| B.4.4.6. | ユーザー更新処理中にエラーが発生した..... | 124 |
| B.4.4.7. | 認証に失敗した..... | 125 |
| B.4.4.8. | リソースへのアクセス・操作権限がない..... | 125 |
| B.5. | SCIM ユーザーの削除..... | 125 |
| B.5.1. | ユーザーリソース識別子取得..... | 125 |
| B.5.2. | ユーザー削除リクエスト..... | 126 |
| B.5.2.1. | SCIM リクエスト..... | 126 |
| B.5.3. | ユーザー削除成功レスポンス..... | 126 |
| B.5.3.1. | SCIM レスポンス..... | 126 |
| B.5.4. | ユーザー削除エラーレスポンス..... | 126 |
| B.5.4.1. | ユーザーリソースが見つからない..... | 126 |
| B.5.4.2. | ユーザーの削除が競合した..... | 126 |
| B.5.4.3. | ユーザー更新処理中にエラーが発生した..... | 127 |
| B.5.4.4. | 認証に失敗した..... | 127 |
| B.5.4.5. | リソースへのアクセス・操作権限がない..... | 127 |
| 付録 C. | サンプル実装例..... | 128 |
| C.1. | TrustBind/Federation Manager を使った OP の実装..... | 128 |
| C.1.1. | 実装概要..... | 128 |
| C.1.2. | OP の実装..... | 129 |
| C.1.2.1. | 認証プラグインの実装..... | 130 |
| C.1.2.2. | 認可プラグインの実装..... | 131 |
| C.2. | OpenAM を使った OP の実装..... | 132 |
| C.2.1. | OpenAM の導入..... | 132 |
| C.2.1.1. | 事前準備..... | 132 |

| | | |
|--------------------------|-----------------------------------|-----|
| C.2.1.2. | Tomcat での TLS の有効化..... | 132 |
| C.2.1.3. | OpenAM のセットアップ..... | 133 |
| C.2.2. | OpenID Provider の設定..... | 133 |
| C.2.2.1. | OP のセットアップ..... | 133 |
| C.2.2.2. | Relaying Party の登録..... | 136 |
| C.3. | Ruby による OP サーバーのスクラッチ実装例..... | 141 |
| C.3.1. | 実装概要..... | 141 |
| C.3.2. | プログラムサンプル..... | 141 |
| C.3.3. | プログラム構成..... | 142 |
| C.3.4. | 動作確認方法..... | 143 |
| C.3.5. | 簡単な OP の仕様説明..... | 145 |
| C.4. | Ruby による RP サーバーのスクラッチ実装例..... | 147 |
| C.4.1. | 実装概要..... | 147 |
| C.4.2. | プログラムサンプル..... | 147 |
| C.4.3. | プログラム構成..... | 148 |
| C.4.4. | 動作確認方法..... | 150 |
| C.5. | mod_auth_openidc を使った RP の実装..... | 153 |
| C.5.1. | インストール..... | 153 |
| C.5.2. | 設定 (httpd.conf)..... | 153 |
| C.5.3. | 実装上の注意..... | 155 |
| C.5.4. | サンプル設定..... | 156 |
| C.6. | Java による SCIM サーバーのスクラッチ実装例..... | 157 |
| C.6.1. | 実装概要..... | 157 |
| C.6.2. | プログラムサンプル..... | 157 |
| C.6.3. | 利用方法..... | 157 |
| C.6.4. | プログラム構成..... | 159 |
| C.6.5. | ポイント..... | 160 |
| 著者一覧..... | | 162 |
| EIWG 技術タスクフォース参加者一覧..... | | 162 |

第1章 はじめに

昨今、企業でのクラウドサービス利用が急速に進んできている。今後もより多くの企業、より多くの業務へとクラウドサービスの利用が拡大することが見込まれる。

クラウドサービスは、インターネット上のあらゆる場所からアクセスができる特性があるため、クラウドサービス上の企業の機密情報を保護するためには、クラウドサービスのアイデンティティ管理と認証機能を、企業のセキュリティポリシーに適合するように実装することが要求される。

セキュリティポリシーが異なる複数のクラウドサービス、および複数の企業が接続され利用される環境の下で、この要求に対応するためには、利用企業の認証システムとクラウドサービスを ID 連携の技術を用いて相互接続、相互運用する方法が必要となる。クラウドサービスの利用拡大にあわせて、企業もクラウドサービスも ID 連携の技術に対応することがますます重要となっている。

当実装ガイドでは、ID 連携技術である OpenID Connect と SCIM をとりあげ、OpenID Connect を使った認証連携 (SSO) と、SCIM を使ったアイデンティティ管理 (ID 管理) により、利用企業の認証システムとクラウドサービスを相互接続、相互運用するための、一般的かつ最小限の実装について解説する。

OpenID Connect は 2014 年 2 月にローンチされた、複数の組織やアプリケーション間でユーザーの認証結果や属性情報をやりとりするための技術仕様である。OpenID Connect は現在コンシューマ IT の分野での利用が急速に拡大しており、今後エンタープライズ IT の分野でも普及が進むものと考えられている。

SCIM は 2015 年 9 月に RFC 化された、アイデンティティデータのプロビジョニングと管理を行なうための技術仕様である。これまでクラウドサービス毎に独自仕様で実装されていたインタフェースが共通の仕様となることで、エンタープライズ IT で求められる ID 管理機能の実装がより容易になる技術として注目されている。

当実装ガイドに沿って実装された企業の認証システムの導入、クラウドサービスの開発と提供が進むことで、企業がクラウドサービスを利用しやすい環境が広がることを期待している。

1.1. 本実装ガイドの構成

本実装ガイドの第 2 章では、これまで OpenID Connect および SCIM に触れたことがない人のために、OpenID Connect と SCIM の仕様の概要を解説する。本実装ガイドを読み進める上で必要となる概念や用語の理解に役立てていただきたい。

第 3 章では、企業がクラウドサービスを利用するケースをとりあげ、利用企業の認証システムとクラウドサービスの連携のモデルについて解説する。連携の全体像を示した後、連携の設定から実際の利用までの流れに沿って、利用企業の認証システムとクラウドサービスのつながりに主眼を置いた技術解説と、実装時に考慮が必要な事項について解説をする。

第 4 章では、クラウドサービス事業者が自社のクラウドサービスで OpenID Connect と SCIM に対応するための実装内容について解説する。

第 5 章では、クラウドサービス利用企業が自社の認証システムで OpenID Connect と SCIM に対応するための実装内容について解説する。

第 4 章、第 5 章ともに、実装すべきエンドポイントとその振る舞いについて、ひとつずつ解説する。また連携先のエンドポイントを呼び出す際の呼び出し方や、そのレスポンスの扱い方についても解説する。

1.2. 本実装ガイドが参照している技術標準

本実装ガイドは、以下の技術標準に基づいて、具体的な実装について解説をしている。

- OpenID Connect Core 1.0 incorporating errata set 1 [OpenID.Core]
- OpenID Connect Back-Channel Logout 1.0 - draft 01 [OpenID.Backchannel]
- System for Cross-domain Identity Management: Core Schema [RFC7643]
- System for Cross-domain Identity Management: Protocol [RFC7644]
- The OAuth 2.0 Authorization Framework [RFC6749]
- JSON Web Signature (JWS) [RFC7515]
- JSON Web Key (JWK) [RFC7517]
- JSON Web Token (JWT) [RFC7519]

当実装ガイドを作成する上で、これらの技術標準との整合性を保つ最大限の努力をしているが、万が一、当実装ガイドと技術標準の間に矛盾がある場合、あるいは技術標準の更新により矛盾が生じた場合については、技術標準の内容に従うべきである。

第2章 標準プロトコル超入門

当実装ガイドは、OpenID Connect および SCIM に関する予備知識を持つ人を対象としている。これまで OpenID Connect や SCIM に触れたことがない人のために、この章では、OpenID Connect と SCIM の仕様の概要を解説する。

2.1. フェデレーション向けプロトコル OpenID Connect

2.1.1. OpenID Connect とは

OpenID Connect は、OpenID 2.0 の次期バージョンとして策定された仕様群である。OpenID Connect Core 1.0 [OpenID.Core] と、複数の選択仕様から構成されている。

同仕様は、HTTP アクセス認可を実現するプロトコルである OAuth 2.0 [RFC6749] 上に、エンドユーザーの「アイデンティティ」情報をやり取りするための層を定義している。

「クライアント」（リライディング・パーティ、以下 RP と表記）は、「認可サーバー」（OpenID プロバイダー、以下 OP と表記）でのエンドユーザーの認証結果を基に、そのユーザーの「アイデンティティ」情報を受け取り、正しいものとして検証することができる。

想定される活用例としては、

- OP にユーザー認証を一元化し、RP 間のシングルサインオンを実現
- RP 側でユーザーのクレデンシャル（パスワードなど）の管理が不要となることによる、セキュリティ向上と管理負荷の低減
- OP からのユーザー属性情報取得による、RP での新規ユーザー登録の容易化
- エンドユーザーの認証と API アクセス認可の一体化によるユーザー利便性の向上

などがある。

2.1.2. OpenID Connect を使った認証フロー

OpenID Connect では認証のフローとして、3つのフローを定義している。

1. 認可コードフロー
2. Implicit フロー
3. ハイブリッドフロー

当実装ガイドでは、一般的に使用される、認可コードフローおよび Implicit フローについて解説する。

2.1.2.1. 認可コードフロー

認可コードフローは、認証要求を受けた OP が「認可コード」と呼ばれる値をエンドユーザー経由で RP に返却し、RP が OP へ「認可コード」を直接提示することで、ID トークン（とアクセストークン）を受け取るフローである。

エンドユーザーが使用しているブラウザやアプリケーションに ID トークンを渡す必要がないため、安全に ID トークンを交換できる方式となる。

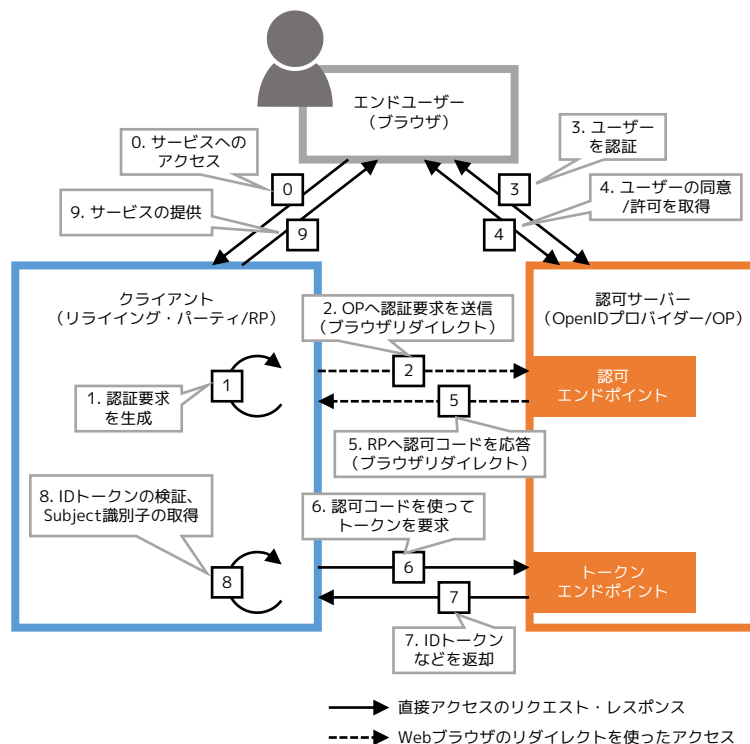


図 2-1: OpenID Connect 認可コードフロー

1. クライアントは必要なパラメータを含む認証要求を生成する。
2. エンドユーザーのアクセスを OP にリダイレクトし、クライアントは OP へ認証要求を送信する。
3. OP はエンドユーザーを認証する。
4. OP エンドユーザーの同意/認可を得る。
5. OP はエンドユーザーのアクセスをクライアントへリダイレクトする。このとき、認可コードをリダイレクト先の URI にクエリパラメータとして付加する。

6. クライアントはトークンエンドポイントへ認可コードをつけてリクエストを送信する。
7. クライアントは ID トークン（とアクセストークン）を受け取る。
8. クライアントは ID トークンを検証し、エンドユーザーの Subject 識別子を取得する。
9. クライアントはエンドユーザーにサービスを提供する。

2.1.2.2. Implicit フロー

Implicit フローは、認証要求を受けた OP が ID トークン（とアクセストークン）を直接エンドユーザー経由で RP に返却するフローである。

ID トークンが、エンドユーザーが使用しているブラウザやアプリケーションを経由して RP に返却されるため、RP では ID トークンが改ざんされていないことの検証を行なうことが重要である。

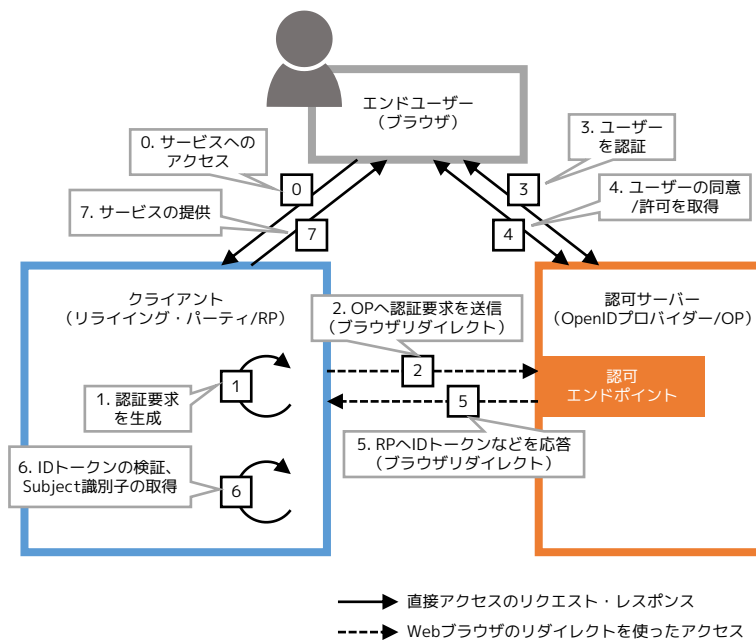


図 2-2: OpenID Connect Implicit フロー

1. クライアントは必要なパラメータを含む認証要求を生成する。
2. エンドユーザーのアクセスを OP にリダイレクトし、クライアントは OP へ認証要求を送信する。
3. OP はエンドユーザーを認証する。
4. OP エンドユーザーの同意/認可を得る。

5. OP はエンドユーザーのアクセスをクライアントへリダイレクトする。このとき、ID トークン（とアクセストークン）をリダイレクト先の URI にフラグメントとして付加する。
6. クライアントは ID トークンを検証し、エンドユーザーの Subject 識別子を取得する。
7. クライアントはエンドユーザーにサービスを提供する。

また、Implicit フローでは、RP が OP へ直接アクセスすることなく ID トークンを受け取ることができる点も重要な特徴である。

2.1.3. ID トークン

OpenID Connect では、OP でのエンドユーザーの認証結果を ID トークンとして RP へ返却する。ID トークンは、認可サーバーによるエンドユーザーの認証についての情報（クレーム）を含むセキュリティトークンである。

ID トークンは JWT (JSON Web Token) [RFC7519] の形式で提供され、JWS (JSON Web Signature) [RFC7515] の仕様に従って署名されていなければならない。また、JWS と JWE (JSON Web Encryption) [RFC7516] を用いて署名をつけた上で暗号化が行なわれることもある。

ID トークンには、次のクレームが含まれる。

1. iss

(必須) ID トークンの発行者 (Issuer) を表す識別子。https スキームを使用した大文字と小文字を区別する URL である。

2. sub

(必須) 認証された主体 (ユーザー) を表す Subject 識別子。発行者内で一意であり決して再割り当てされない識別子である。大文字と小文字を区別する、255 文字以下の ASCII 文字列でなければならない。

3. aud

(必須) ID トークンの使用を意図しているクライアントを表す。複数の client_id からなる配列か、もしくは 1 つの client_id を値として持つ。

4. exp

(必須) ID トークンの有効期限。UTC で計測される 1970-01-01T0:0:0Z からの秒数。

5. iat

(必須) ID トークンが発行された時刻。UTC で計測される 1970-01-01T0:0:0Z からの秒数。

6. auth_time

エンドユーザーの認証が行なわれた時刻。UTC で計測される 1970-01-01T0:0:0Z からの秒数。リクエストパラメータで `max_age` パラメータが渡されたとき、または `auth_time` が不可欠クレームとして要求されたときは、`auth_time` クレームは必須である。そうでない場合は、`auth_time` クレームはオプションである。

7. nonce

リプレイ攻撃の軽減のため、ID トークンとクライアントセッションの関連付けるに使われる文字列。認証リクエストに `nonce` が含まれていた場合、OP はその値を ID トークンに `nonce` クレームとして含めなければならない。

8. acr

(オプション) 認証コンテキスト・クラス・リファレンス (Authentication Context Class Reference)。OP の実施したユーザー認証が、どの認証コンテキスト・クラスに属するかを示す。

9. amr

(オプション) 認証手段リファレンス (Authentication Methods References)。OP が実施したユーザー認証の手段を示す。

10. azp

(オプション) `aud` に認証を要求した RP の `client_id` と異なる値を返すとき、`azp` に認証を要求した RP の `client_id` を含めなければならない。

2.1.4. ユーザーの属性情報の受け渡し方法

OpenID Connect では、認証要求を行なう際にユーザーの属性情報を要求することで、エンドユーザーの同意/許可の下、必要な属性情報を受け取る方法が定められている。この仕組みは、RP での新規ユーザー登録を行なう場合などに活用される。

2.1.4.1. ユーザーの属性情報の要求

RP は認証要求のパラメータに取得したいユーザー属性を指定することで、ユーザーの属性情報を取得することができる。指定方法には次の方法がある。

1. `scope` パラメータに、OP が定義した「ユーザー属性のセット」の名称を用いて指定する方法
2. `claims` パラメータに、リクエストするユーザー属性名を個々に指定する方法

ここでは実環境において広く使用されている `scope` パラメータを用いて指定する方法について解説する。

認証要求の `scope` パラメータは OAuth 2.0 で定義されたものであるが、OpenID Connect では要求するユーザー属性のセットを指定するためにも用いられる。OpenID Connect では以下の `scope` 値を定義している。

1. profile

(オプション) ユーザーの既定のプロフィールクレームを要求する。既定のプロフィールクレームには次のものが含まれる。

`name` (氏名)、`family_name` (姓)、`given_name` (名)、`middle_name` (ミドルネーム)、`nickname` (ニックネーム)、`preferred_username` (エンドユーザーが希望するユーザー名)、`profile` (プロフィールページの URL)、`picture` (プロフィール画像の URL)、`website` (ユーザーの Web サイトの URL)、`gender` (性別)、`birthdate` (誕生日)、`zoneinfo` (タイムゾーン)、`locale` (言語)、`updated_at` (更新日時)

2. email

(オプション) メールアドレスに関するクレームを要求する。次のクレームが含まれる。

`email` (メールアドレス)、`email_verified` (メールアドレスが検証済みかどうか)

3. address

(オプション) `address` クレームを要求する。`address` クレームは JSON オブジェクトの形式である。

4. phone

(オプション) 電話番号に関するクレームを要求する。次のクレームが含まれる。

`phone_number` (電話番号)、`phone_number_verified` (電話番号が検証済みかどうか)

これらのあらかじめ定義されている値以外に、OP が独自に定義した「ユーザー属性のセット」の名称を指定することもできる。

2.1.4.2. ユーザーの属性情報の受け取り

OpenID Connect では RP から要求されたユーザー属性情報を連携する方法として、ID トークンに属性情報を含める方法と、RP からアクセス可能な UserInfo エンドポイントを通じて提供する方法の 2 つを定義している。

2.1.4.2.1. ID トークンに含める方法

ID トークンには [2.1.3 節] に示したクレーム以外のクレームも含めることができる。たとえば、Google Identity Platform では、scope として email を要求した場合に、email、email_verified クレームが ID トークンに含まれて応答される。

```
{
  "iss": "accounts.google.com",
  "at_hash": "HK6E_P6Dh8Y93mRNtsDB1Q",
  "email_verified": true,
  "sub": "10769150350006150715113082367",
  "azp": "1234987819200.apps.googleusercontent.com",
  "email": "jsmith@example.com",
  "aud": "1234987819200.apps.googleusercontent.com",
  "iat": 1353601026,
  "exp": 1353604926,
  "hd": "example.com"
}
```

(<https://developers.google.com/identity/protocols/OpenIDConnect#obtainuserinfo> より引用)

しかし、scope で要求されたクレームを ID トークンで応答する実装例は限定的であり、Google Identity Platform でも email 関連のクレームに限定されている。

2.1.4.2.2. UserInfo エンドポイントを通じて提供する方法

UserInfo エンドポイントは、OpenID Connect 仕様に定義されている OP が RP にユーザー情報を提供するための API である。OAuth 2.0 の保護されたリソース (Protected Resource) として提供され、RP は認証要求の際に ID トークンと同時に取得したアクセストークンを用いてアクセスする。OP はユーザー情報を、通常は JSON 形式にて RP に返却する。

以下は、OpenID Connect 仕様に記述されている、UserInfo エンドポイントへのリクエスト・レスポンスの例である。

```
GET /userinfo HTTP/1.1
Host: server.example.com
Authorization: Bearer SIAV32hkKG

HTTP/1.1 200 OK
Content-Type: application/json

{
  "sub": "248289761001",
  "name": "Jane Doe",
  "given_name": "Jane",
  "family_name": "Doe",
  "email": "janedoe@example.com",
  "picture": "http://example.com/janedoe/me.jpg"
}
```

2.1.5. セッション管理に関する関連仕様

OpenID Connect の中核となる仕様にはセッション管理に関する仕様は定義されていないが、現在、実装者向けドラフト仕様あるいはドラフト仕様のレベルで、セッション管理に関する次の3つの仕様の策定が進められている。

- OpenID Connect Back-Channel Logout 1.0 [OpenID.Backchannel]
- OpenID Connect Session Management 1.0 [OpenID.Session]
- OpenID Connect HTTP-Based Logout 1.0 [OpenID.Logout]

このうち、当実装ガイドでは OpenID Connect Back-Channel Logout に基づいた実装について扱う。OpenID Connect Back-Channel Logout は、OP と RP が直接通信を行なうことで、OP が RP にログアウトを要求し、RP のセッションをログアウトさせることができる仕様となっている。他の 2 つの仕様がブラウザを介する方式であることに対し、OpenID Connect Back-Channel Logout はブラウザを介する必要がない点が特徴である。

2.2. プロビジョニング向けプロトコル SCIM

2.2.1. SCIM とは

SCIM (System for Cross-domain Identity Management) はクラウドベースのアプリケーションにおけるアイデンティティ管理とサービスをより簡単に構築できるように設計された仕様である。SCIM は HTTP プロトコル (REST API) と JSON を利用してクラウドアプリケーション間のアイデンティティデータのプロビジョニングと管理を行なう。

現在のクラウドアプリケーションに対するアイデンティティデータのプロビジョニングと管理の実装は、クラウドアプリケーションごとにクラウドサービスプロバイダーから提供されている

独自インタフェース（独自 API、独自レイアウトの CSV ファイルなど）を用いて実装されている。しかし、今後ますますエンタープライズ分野でクラウドアプリケーションが活用されるようになる上で、利用する企業が数多くのクラウドアプリケーションごとに異なるインタフェースを用いてアイデンティティデータのプロビジョニングと管理を実装しなければならないのは大きな足かせとなる。

SCIM は、様々なクラウドアプリケーションに対するアイデンティティデータのプロビジョニングと管理を実装するための共通インタフェースの標準仕様を提供することにより、この問題を解決することを目的としている。

SCIM は、クラウドベースの HTTP Web アプリケーションである「サービスプロバイダー」と、サービスプロバイダーに対しアイデンティティデータのプロビジョニングと管理を行なう Web サイトまたはアプリケーションである「クライアント」間における、共通のスキーマとプロトコルと提供する。

2.2.2. SCIM で扱えるデータ

SCIM Core Schema [RFC7643] では、SCIM Protocol が扱うデータの構造として、6つのリソースのスキーマを定義している。

| No | リソース名 | 説明 |
|----|------------------------|---|
| 1 | ユーザー | ユーザーを表現するための属性を持つリソーススキーマ（例えば、名前や email など） |
| 2 | グループ | グループを表現するための属性を持つリソーススキーマ |
| 3 | Enterprise User スキーマ拡張 | 企業のユーザーを表現するためのユーザースキーマの拡張 |
| 4 | サービスプロバイダー設定 | サービスプロバイダーにおける SCIM 仕様の対応状況を説明するスキーマ |
| 5 | リソースタイプ | リソースタイプのエンドポイントや拡張状態を説明するスキーマ |
| 6 | スキーマ定義 | リソーススキーマの属性のタイプや必須要件を説明するスキーマ |

当実装ガイドでは、ユーザーリソーススキーマ、Enterprise User スキーマ拡張、スキーマ定義のためのスキーマ（スキーマの拡張方法）を扱うため、以下ではこの 3つのリソーススキーマについて解説する。

2.2.2.1. SCIM で扱える属性の型

SCIM Core Schema では、SCIM の属性として扱える型が定義されている。

SCIM の属性として扱える型には以下の 8つがある。

| No | 型 | 説明 |
|----|-----------|--|
| 1 | String | UTF-8 でエンコードされた文字列 |
| 2 | Boolean | 真偽値。true もしくは false のいずれか。 |
| 3 | Decimal | 実数 |
| 4 | Integer | 整数 |
| 5 | DateTime | 日時を表す文字列。[XML-Schema]で dateTime 型として定義されたフォーマットを用いる。 |
| 6 | Binary | Base64 エンコードされたバイナリデータ |
| 7 | Reference | リソースを指す URI |
| 8 | Complex | 1つ以上の単純型をサブ属性に持つ値 |

また、SCIM の属性は取る値の数により、Singular 属性と Multi-Valued 属性の 2 種類に分類される。

1. Singular 属性

属性の値として、0 個もしくは 1 個の値をとる属性を指す。

2. Multi-Valued 属性

属性の値として、0 個以上の値を取る属性を指す。

Multi-Valued 属性は、JSON のリスト形式で表現される。リストの要素は、String や Integer といった単純型か、Complex 型かのいずれかとなる。

Complex 型の値を用いる場合、Multi-Valued 属性向けに定義された次のサブ属性を含めることができる。

| No | サブ属性名 | 型 | 説明 |
|----|---------|-----------|---|
| 1 | type | String | 属性値を識別するラベル。メールアドレスの場合 "work" や "home" などの値を取り得る。 |
| 2 | primary | Boolean | 属性値のうち優先的に使用する属性を表すフラグ |
| 3 | display | String | 表示に用いる名称 |
| 4 | value | | 属性の値。その属性に適した型を用いることができるが、Complex 型は使用できない。 |
| 5 | \$ref | Reference | 属性がリファレンスの場合、その属性を表す URI を格納する。 |

2.2.2.2. ユーザーリソーススキーマ

ユーザーリソーススキーマは、ユーザーを表現するための属性（例えば、名前や email など）を持つリソースを定義する。ユーザーリソーススキーマでは、以下の属性が定義されている。

- 共通属性

| No | 属性名 | サブ属性名 | 型 | 説明 |
|----|------------|--------------|-----------|--------------------|
| 1 | id | | String | リソース識別子（読み取り専用属性） |
| 2 | externalId | | String | クライアント側（企業側）での識別子 |
| 3 | meta | | Complex | メタデータ（読み取り専用属性） |
| 4 | | resourceType | String | リソースタイプ名 |
| 5 | | created | DateTime | 作成した時刻 |
| 6 | | lastModified | DateTime | 最後に更新した時刻 |
| 7 | | location | Reference | リソースにアクセスするための URI |
| 8 | | version | String | リソースのバージョン (ETag) |

- ユーザーリソースの属性

| No | 属性名 | サブ属性名 | 型 | MV | 説明 |
|----|-------------------|-----------------|-----------|----|---------------------------------------|
| 1 | userName | | String | | ユーザー名（ユニーク ID） |
| 2 | name | | Complex | | 名前 |
| 3 | | formatted | String | | フルネーム |
| 4 | | familyName | String | | 姓 |
| 5 | | givenName | String | | 名 |
| 6 | | middleName | String | | ミドルネーム |
| 7 | | honorificPrefix | String | | プレフィックス。Mr.など。 |
| 8 | | honorificSuffix | String | | サフィックス。三世、など。 |
| 9 | displayName | | String | | 表示名 |
| 10 | nickName | | String | | ニックネーム |
| 11 | profileUrl | | Reference | | プロフィールが参照できる URL |
| 12 | title | | String | | 役職 |
| 13 | userType | | String | | 職種 |
| 14 | preferredLanguage | | String | | 使用言語。Accept-Language と同じ。 ("en-US"形式) |

| | | | | | |
|----|--------------|---------------|---------|---|--|
| 15 | locale | | String | | 地域 ("en-US"形式) |
| 16 | timezone | | String | | タイムゾーン |
| 17 | active | | Boolean | | 管理状態 (true ならログイン可、false なら不可など) |
| 18 | password | | String | | パスワード (書き込み専用属性) |
| 19 | emails | | Complex | ○ | メールアドレス のリスト |
| 20 | | value | String | | メールアドレス |
| 21 | | display | String | | 表示用名称 |
| 22 | | type | String | | "work", "home", "other" のラベル |
| 23 | | primary | Boolean | | 優先的に使用するメールアドレスのフラグ |
| 24 | phoneNumbers | | Complex | ○ | 電話番号 のリスト |
| 25 | | value | String | | 電話番号 |
| 26 | | type | String | | "work", "home", "mobile", "fax", "pager", "other" のラベル |
| 27 | | primary | Boolean | | 優先的に使用する電話番号のフラグ |
| 28 | ims | | Complex | ○ | インスタントメッセージのアドレス (ICQ, Skype など) のリスト |
| 29 | photos | | Complex | ○ | ユーザーの写真の URL のリスト |
| 30 | addresses | | Complex | ○ | 住所 のリスト |
| 31 | | formatted | String | | 表示用に整形された住所 |
| 32 | | streetAddress | String | | 番地、建物名など |
| 33 | | locality | String | | 市区町村 |
| 34 | | region | String | | 都道府県 |
| 35 | | postalCode | String | | 郵便番号 |
| 36 | | country | String | | 国 ("US", "JP" などの短縮形式) |
| 37 | | type | String | | "work", "home", "other" のラベル |
| 38 | | primary | Boolean | | 優先的に使用する住所のフラグ |

| | | | | | |
|----|------------------|--|---------|---|---------------------|
| 39 | groups | | Complex | ○ | ユーザーが所属するグループのリスト |
| 40 | entitlements | | Complex | ○ | ユーザーが持っている資格・権利のリスト |
| 41 | roles | | Complex | ○ | ユーザーのロールのリスト |
| 42 | x509Certificates | | Complex | ○ | ユーザーの電子証明書 のリスト |

- 表中、MV 欄に「○」が示された属性は、Multi-Valued 属性である。

2.2.2.3. Enterprise User スキーマ拡張

Enterprise User スキーマ拡張は、企業のユーザーを表現するために必要な属性を定義し、ユーザースキーマを拡張するものとして使用する。Enterprise User スキーマ拡張では、以下の属性が定義されている。いずれも Singular 属性である。

| No | 属性名 | サブ属性名 | 型 | 説明 |
|----|----------------|-------------|-----------|---------------------------|
| 1 | employeeNumber | | String | 従業員番号 |
| 2 | costCenter | | String | 原価部門 |
| 3 | organization | | String | 組織（企業） |
| 4 | division | | String | 組織内の部門（事業部など） |
| 5 | department | | String | 組織内の部門（部署など） |
| 6 | manager | | Complex | 上司 |
| 7 | | value | String | 上司ユーザーのリソース識別子 |
| 8 | | \$ref | Reference | 上司ユーザーのリソースにアクセスするための URI |
| 9 | | displayName | String | 上司ユーザーの表示名（読み取り専用属性） |

2.2.2.4. スキーマ定義のためのスキーマ

SCIM では、スキーマを JSON 形式で表現するためのスキーマが用意されている。スキーマを拡張する場合は、スキーマ定義のためのスキーマを用いて、スキーマを定義することができる。

スキーマ定義のためのスキーマでは、以下の属性が定義されている。

| No | 属性名 | サブ属性名 | 説明 |
|----|-------------|-----------------|---|
| 1 | id | | スキーマを識別する URI |
| 2 | name | | スキーマ名 |
| 3 | description | | 説明 |
| 4 | attributes | | 定義する属性 のリスト |
| 5 | | name | 属性名 |
| 6 | | type | 属性の型。string, boolean, decimal, integer, dateTime, refernece, complex のいずれか。 |
| 7 | | subAttributes | complex 型の属性が持つサブ属性 のリスト |
| 8 | | multiValued | Multi-Valued 属性のフラグ |
| 9 | | description | 属性の説明 |
| 10 | | required | 必須属性のフラグ |
| 11 | | canonicalValues | 推奨値のリスト |
| 12 | | caseExact | string 型の属性を比較する場合の大文字小文字区別のフラグ |
| 13 | | mutability | 属性値の変更可否の指定。readOnly, readWrite, immutable, writeOnly のいずれか。 |
| 14 | | returned | レスポンスに属性を返すかどうかの指定。always, never, default, request のいずれか。 |
| 15 | | uniqueness | 値の一意性の指定。none, server, global のいずれか。 |
| 16 | | referenceType | reference 型の属性が参照するリソースタイプ |

当実装ガイドでは、EIWG 拡張スキーマを定義するために使用している。

2.2.3. SCIM を使ったデータ操作

SCIM Protocol [RFC7644] では、SCIM Core Schema で定義されたデータ（リソース）を操作する REST ベースのプロトコルを定義している。

2.2.3.1. HTTP メソッドと操作

SCIM では次の HTTP メソッドを用いて、リソースの操作を行なえる。

| No | HTTP メソッド | 行なう操作 |
|----|-----------|-------------------------------------|
| 1 | GET | リソースを取得する |
| 2 | POST | リソースを作成する。検索リクエスト、リソースのバルク操作にも使用する。 |
| 3 | PUT | 置き換えによりリソースを変更する |
| 4 | PATCH | 部分更新によりリソースを変更する |
| 5 | DELETE | リソースを削除する |

2.2.3.2. リソース（データ）に対する操作

リソースを操作する場合は、リソースに対応するエンドポイントへ、行ないたい操作に対応する HTTP メソッドでリクエストを送る。

| No | リソース | エンドポイント名 | 行なう操作 | 説明 |
|----|--------------|------------------------|-------------------------------|---------------------------|
| 1 | ユーザー | /Users | GET, POST, PUT, PATCH, DELETE | ユーザーの取得、作成、変更、削除を行なう |
| 2 | グループ | /Groups | GET, POST, PUT, PATCH, DELETE | グループの取得、作成、変更、削除を行なう |
| 3 | 自分自身 | /Me | GET, POST, PUT, PATCH, DELETE | 認証された主体（ユーザーなど）のリソースを操作する |
| 4 | サービスプロバイダー設定 | /ServiceProviderConfig | GET | サービスプロバイダーの設定情報を取得する |
| 5 | リソースタイプ | /ResourceTypes | GET | 対応しているリソースタイプの情報を取得する |
| 6 | スキーマ定義 | /Schemas | GET | 対応しているスキーマ定義の情報を取得する |
| 7 | バルク操作 | /Bulk | POST | リソースに対するバルク操作をリクエストする |
| 8 | 検索 | [prefix]/.search | POST | リソースの検索をリクエストする |

2.2.3.3. エンドポイントアクセス時の認証方法

クライアントがサービスプロバイダーの SCIM エンドポイントにアクセスをする場合には、適切な認証を行なう必要がある。この認証方式は SCIM の仕様では定義されておらず、TLS や HTTP の認証スキームを使用することになる。

SCIM Protocol では、下記のような認証方式の利用が想定されている。

| No | 認証方式 | 説明 |
|----|--------------|---|
| 1 | TLS クライアント認証 | TLS でクライアント証明書を使用する認証方式 |
| 2 | HOBA 認証 | HTTP Origin-Bound Authentication (RFC7486) による JavaScript ベースの公開鍵認証方式 |
| 3 | ベアラートークン | OAuth 2.0 認可フローで発行されるベアラートークンを使用する方式 |
| 4 | POP トークン | トークンの所有者を、所有者と認可サーバー間の鍵交換と署名検証によって証明する方式 |
| 5 | Cookie | 認証状態を表す Cookie を用いる方式 |
| 6 | Basic 認証 | Basic 認証を用いる方式。単体の使用ではなく他の要素と組み合わせて使用する。 |

サービスプロバイダーは、アクセス元のクライアントを適切に認証し、クライアントが持つリソースの読み取りや書き込みの操作の権限を適切に判定した上で、リクエストを処理しなければならない。

2.2.4. SCIM 1.1 から SCIM 2.0 への変更点の要点

SCIM は 2011 年 12 月にバージョン 1.0 がリリースされ、2012 年 7 月にバージョン 1.1 がリリースされた。これらの仕様は、Open Web Foundation (OWF) 下で標準化が進められた。SCIM バージョン 2.0 は、Internet Engineering Task Force (IETF) 下で標準化が進められ、2015 年 9 月に RFC 化された。

旧バージョンである SCIM 1.1 と最新の SCIM 2.0 では、重要な変更点がいくつか存在するため、下記に要点を示す。

- データフォーマットの変更

SCIM 1.1 ではデータフォーマットとして JSON と XML をサポートしていたが、SCIM 2.0 では JSON のみをサポートするようになった。

- 用語の変更

SCIM 1.1 では「サービスプロバイダー（クラウドアプリケーションを提供する側、例えば Salesforce.com など）」に対し、利用者側を「コンシューマ」と定義していたが、SCIM 2.0 では利用者側を「クライアント」と定義している。

- アプリケーション間の認証・認可

SCIM 1.1 ではアプリケーション間の認証・認可として、OAuth 2.0 Bearer Token [RFC6750] を推奨していたが、SCIM 2.0 では TLS クライアント認証や HOBA 認証 (HTTP Origin-Bound Authentication) などのいくつかの選択肢を示している。

- セキュリティ考察の拡充

SCIM 1.1 に比べ、SCIM 2.0 ではプロトコルとサービスプロバイダーでのセキュリティ対策について具体的な言及がされている。SCIM では、個人情報やセンシティブ情報、パスワードなどのデータを扱うため、実装時のセキュリティ対策が促されている。

- エンドポイントの変更

SCIM 1.1 ではサービスプロバイダーの /Schemas エンドポイントにおいて、クラウドアプリケーションで提供されているエンドポイントやスキーマ属性の説明を取得できる仕様であったが、SCIM 2.0 では、/ResourceTypes エンドポイントで各エンドポイントの説明を、/Schemas エンドポイントでスキーマ属性の説明を取得できるように変更された。

また、認証されたサブジェクトのエイリアスとしての /Me エンドポイントや、POST による検索を提供する /.search エンドポイントの定義が追加された。

第3章 モデルユースケースと実装の概要

3.1. 利用企業の認証システムとクラウドサービスの連携モデル

利用企業の認証システムとクラウドサービスの連携として、以下のような一般的なケースを取り上げ、実装方法を解説する。

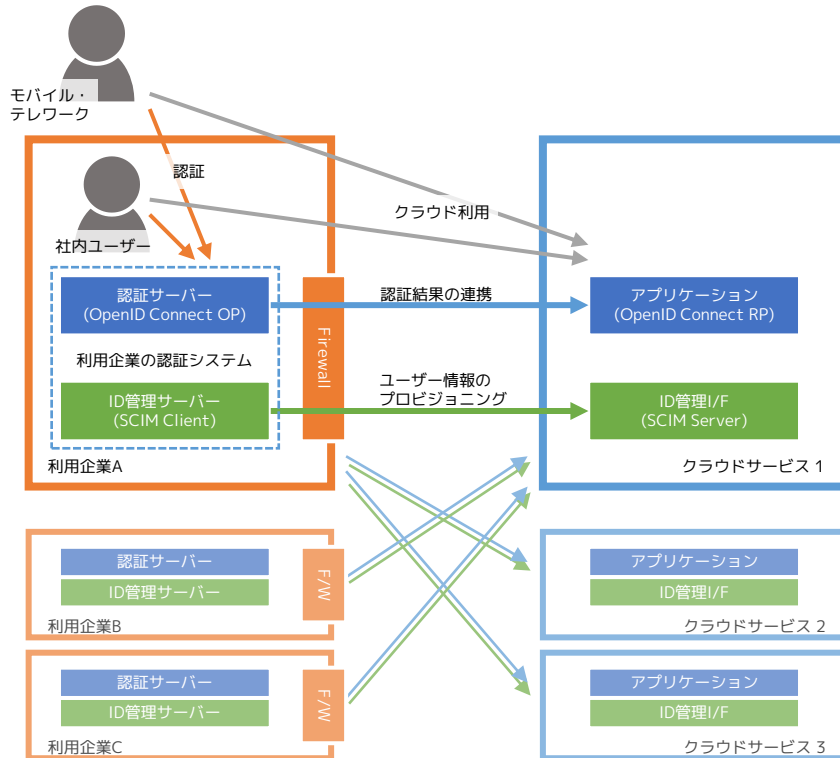


図 3-1: 利用企業の認証システムとクラウドサービスの連携

クラウドサービスの利用企業は認証サーバー（OpenID Connect OP）と ID 管理サーバー（SCIM Client）からなる認証システムを実装する。当実装ガイドでは、この認証システムがファイアウォール内に設置されていることを想定する。

クラウドサービスはアプリケーション（OpenID Connect RP）と ID 管理のためのインタフェース（SCIM Server）を提供する。

クラウドサービスの利用企業は ID 管理サーバーより、SCIM プロトコルを用いて、定期的にクラウドサービスへユーザー情報のプロビジョニングを行なう。

利用企業のユーザーは、利用企業の認証サーバーにより認証され、その認証結果が OpenID Connect を用いてクラウドサービスへ連携されることで、クラウドサービスが提供するアプリケーションを利用できる。

クラウドサービスは複数の利用企業へサービスを提供する。利用企業は複数のクラウドサービスを利用しうる。

クラウドサービスは、必要とするユーザーや組織の情報のレベルにより、大きく 3 つのタイプに分類することができる。

1. シンプルなユーザー情報を要求するクラウドサービス

このタイプのクラウドサービスでは、主に表示に用いることを目的に、ユーザーの氏名や組織に関する情報が必要となる。

2. 多言語対応されたユーザー情報を要求するクラウドサービス

このタイプのクラウドサービスには、言語により表示を切り替える必要があるものや、名前の「よみがな」の提供が必要なものが分類される。

特に「よみがな」は、ユーザー情報を 50 音順に並べなおして表示場合や、ユーザー情報をアドレス帳として提供する場合に必要となる要件である。

3. 日本特有の多階層の組織情報やそれに対応する役職の情報を要求するクラウドサービス

ワークフローやグループウェアなどのアプリケーションの場合、組織単位である「事業部」、「部」、「課」や「本社」、「支社」などによる、多階層の組織構造の情報が必要となることがある。

また、ユーザーが複数の組織に属す、いわゆる兼務をしている場合には、どの組織でどの役職であるかという情報が必要となる場合もある。

このような場合には、ユーザー情報のほか、組織情報や役職情報がクラウドサービスへプロビジョニングされている必要がある。

当実装ガイドでは、利用企業の認証システムとクラウドサービスの一般的な連携として上記の 1., 2. のケースに対応することをターゲットとし、実装方法を解説する。

3.2. 連携のための初期設定

利用企業の認証システムとクラウドサービスを連携するための設定は、次の流れで実施する。

1. クラウドサービスが設定に必要な基本情報を利用企業へ提供する。
2. 利用企業の ID 管理サーバーで SCIM クライアントとしての設定を行なう。
3. 利用企業の認証サーバーへクラウドサービスを OpenID Connect RP として登録する。
4. クラウドサービスに認証サーバーを OpenID Connect OP として登録する。

クラウドサービスは、連携に必要な情報提供と設定のための専用の管理画面を提供することで、利用企業が利用を開始しやすい環境を作ることができる。

3.2.1. クラウドサービスが設定に必要な基本情報を提供する

クラウドサービスは利用企業へ次の情報を提供する。

| No | 情報 | 例 |
|----|----------------------------------|---|
| 1 | SCIM User エンドポイントの URI | https://scim.svc.example.net/User https://scim.svc.example.net/v2/User |
| 2 | SCIM 検索エンドポイントの URI | https://scim.svc.example.net/.search https://scim.svc.example.net/v2/.search |
| 3 | SCIM アクセス用の認証情報 | userid: company123456 password: gyN9JR7zTX4mywt4qb1nYuHiM07XR4CU |
| 4 | OpenID Connect リダイレクト先 URI の一覧 | https://www.svc.example.net/cb https://www.svc.example.net/cb2 |
| 5 | OpenID Connect ログアウトエンドポイントの URI | https://www.svc.example.net/bc_logout |
| 6 | サービス利用時のユーザー名の指定 | メールアドレスを使用する |

クラウドサービスの SCIM サーバーでは、ユーザー情報のプロビジョニング用に User エンドポイント、検索エンドポイントを実装し、利用企業へ提供する。SCIM Protocol Versioning に対応するために、バージョン番号を含むエンドポイントとバージョン番号を含まないエンドポイントの両方を提供する。

クラウドサービスは、OpenID Connect を用いた認証時に `redirect_uri` に指定しうる URI の一覧を利用企業へ提供する。OpenID Connect の仕様により、これらの URI は完全一致での検証が行なわれる。

クラウドサービスは、利用者のセッションをログアウトさせる必要がある場合の通知を受け取るため、ログアウトエンドポイントを実装し、利用企業へ提供する。

クラウドサービスは、サービスを利用するときのユーザー名（クラウドサービスへのログイン名）に何を使用するかを指定しなければならない。一般的には、ユーザーのメールアドレスを使用する方法がある。この値は、利用企業の ID 管理サーバーが決定できる値である必要がある。

3.2.2. 利用企業の ID 管理サーバーで SCIM クライアントしての設定を行なう

利用企業の ID 管理サーバーには、ユーザー情報のプロビジョニングが行なえるよう、SCIM User エンドポイント、検索エンドポイントの URI と、エンドポイントにアクセスするための認証情報を設定する。

| No | 情報 | 例 |
|----|------------------------|---|
| 1 | SCIM User エンドポイントの URI | https://scim.svc.example.net/User https://scim.svc.example.net/v2/User |
| 2 | SCIM 検索エンドポイントの URI | https://scim.svc.example.net/.search https://scim.svc.example.net/v2/.search |
| 3 | SCIM アクセス用の認証情報 | userid: c7654321 password: gyN9JR7zTX4mywt4qb1nYuHiM07XR4CU |

SCIM User エンドポイント、検索エンドポイントの URI の情報として、バージョン番号を含む URI と含まない URI が提供される。通常の設定ではバージョン番号を含まない URI をプロビジョニング先のエンドポイントとして設定する。

SCIM エンドポイントアクセス時の通信路は、TLS による暗号化が行なわれている必要がある。

SCIM エンドポイントアクセス時の認証方式として、少なくとも Basic 認証に対応する。Basic 認証に用いるパスワードは、クラウドサービスが十分な長さ（32 バイト以上）を持つパスワードを発行したものを使用することが推奨される。このパスワードは、クラウドサービスが提供する利用企業の管理者向けの管理画面で照会する。

クラウドサービスは、SCIM エンドポイントにアクセスする SCIM クライアントの IP アドレスを用いたアクセス制御、およびアクセス記録の提供を実装することが推奨される。

3.2.3. 利用企業の認証サーバーへクラウドサービスを OpenID Connect RP として登録する

利用企業の認証サーバーからクラウドサービスへ、OpenID Connect の Implicit フローを用いて認証連携が行なえるよう、クラウドサービスを RP として登録する。

クラウドサービスから提供される情報は、リダイレクト先 URI の一覧と、ログアウト要求を受け付けるエンドポイントの URI の 2 つである。

| No | 情報 | 例 |
|----|----------------------------------|---|
| 1 | OpenID Connect リダイレクト先 URI の一覧 | https://www.svc.example.net/cb https://www.svc.example.net/cb2 |
| 2 | OpenID Connect ログアウトエンドポイントの URI | https://www.svc.example.net/bc_logout |

認証サーバーへの登録により、クラウドサービスの `client_id` が確定する。

| No | 情報 | 例 |
|----|------------------------|------------|
| 1 | <code>client_id</code> | pWBoRam9sG |

3.2.4. クラウドサービスに認証サーバーを OpenID Connect OP として登録する

利用企業の認証サーバーからクラウドサービスへ、OpenID Connect の Implicit フローを用いて認証連携が行なえるよう、クラウドサービスに認証サーバーを OP として登録する。

登録に必要な利用企業の認証サーバーの情報は以下のとおりである。

| No | 情報 | 例 |
|----|---------------------------------------|--|
| 1 | クラウドサービスに割り当てた <code>client_id</code> | pWBoRam9sG |
| 2 | Issuer の識別子 | https://op.com.example.co.jp |
| 3 | 認証エンドポイント URI | https://op.com.example.co.jp/authorize |
| 4 | ID トークン署名検証用の公開鍵 | PEM 形式 or JWK 形式の公開鍵 |
| 5 | ID トークン署名検証用の公開鍵の ID | iAw5 |

クラウドサービスは、利用企業の管理者向けの機能として認証サーバーの登録画面を提供し、これらの情報をフォームに記入して登録できるようにしておかなければならない。

ID トークンの署名を検証するための公開鍵の登録方法として、PEM 形式の公開鍵を登録する方法と、JWK (JSON Web Key) [RFC7517] 形式の公開鍵を登録する方法のいずれか、もしくは両方に対応しなければならない。

署名検証の際には、使用する公開鍵を `kid` (Key ID) を用いて識別する必要があるため、PEM 形式で公開鍵を登録する場合は、`kid` (Key ID) を併せて入力できなければならない。

- PEM 形式の公開鍵の例

```
-----BEGIN PUBLIC KEY-----
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAKerCvNwaqXNtkm1+pSZm
LIVxnG2mvTGJhX6ROGwkjeqFz0w+Q/iJRkBbb1wyGbZIVIf01CafH2utjLm6+Op8
p8m9EGJMsM0nR1/d7hheWnKIBpm08IWFavXVXRWb6VBOK9A8GjDzgOriyF8E9CIa
aZpnGEbdZM1CER53izfOD0b4hk7d8b32vcP8nJYzUwM0PEYyXf8F80M80ijDVbKn
S2Lj+OHDjHmzcsw1oWdnpnJfs5i4xsfe8GA0uE5niUseY3Uy7w11uz0+xR4IuJ+0
sTDKN5Za3iEjZS2KkAlnfwPwZYM0ibhZxaqrPzw19mVHLIWs0v79n1i5CFdUrwX2
JwIDAQAB
-----END PUBLIC KEY-----
```

- JWK 形式の公開鍵の例

```
{
  "kty": "RSA",
  "alg": "RS256",
  "kid": "iAw5",
  "n": "26T2mSFqWMIdb4hOBFTQSfIHD6sZNAphQNR6qzgd-xkY6bPfftXs0K41
yJ41AwnpMi0sZXYj_dx17L2sJxJQ6R-q8xwMhj-oc9gTLzNK8EF-FwysQjT3mca1
QjYd7-7Tjr9CdytU0dJaF-nxJs0w8Ck519WKRHgLtFzwEYUeKERnj0tR8jLfn4Dp
LT1N8bdUgV7nL_d2qpSnuxv83SSPzmuyoq91Xv01TFqHyyK1-fzvqAoPhLMT-72S
mev2jNnS2WdDPrRgHvhNsU9XbUoBJswN5yUVFKgprHB2SLntZAmx9L3YZVLFSGzH
kE2ycZrg1_p0JjRDSwcw",
  "e": "AQAB"
}
```

利用企業が、公開鍵を JWK Set 形式で外部からアクセスできる URI (`jwtks_uri`) へ配置できる場合は、公開鍵の交換に `jwtks_uri` を用いても良い。

クラウドサービスが `jwtks_uri` を使った公開鍵の取得に対応することで、利用企業の署名用鍵のローテーション運用負担を下げることができる。

3.3. ユーザー情報のプロビジョニング

クラウドサービスを利用するユーザーの情報は、利用企業の ID 管理サーバーからクラウドサービスへ、事前にかつ定期的にプロビジョニングを行なう。

3.3.1. ユーザー属性

ユーザー情報のプロビジョニングで必要とするユーザー属性を次のとおり定義する。

| No | 属性 | 例 | SCIM 属性名 | サブ属性名 | 型 | M | V | 必須 |
|----|-------------------|-------------------------------|-----------------------|------------|---------|---|---|----|
| 1 | ID 管理サーバー上の識別子 | e1234567 | externalId | | String | | | ○ |
| 2 | サービス利用時のユーザー名 | taro.nippon@com.example.co.jp | userName | | String | | | ○ |
| 3 | 企業 OP でのログイン ID | e1234567 | externalUserName (*2) | | String | | | ○ |
| 4 | ID トークンの識別子 | — | idTokenClaims (*2) | | Complex | ○ | | ○ |
| 5 | Issuer 識別子 | https://op.com.example.co.jp | | issuer | String | | | ○ |
| 6 | Subject 識別子 | e1234567 | | subject | String | | | ○ |
| 7 | 従業員番号 | e1234567 | employeeNumber (*1) | | String | | | |
| 8 | 名前 (母国語) (*3) | — | name | | Complex | ○ | | |
| 9 | 名前 | 日本 太郎 | | formatted | String | | | |
| 10 | 名前 (姓) | 日本 | | familyName | String | | | |
| 11 | 名前 (名) | 太郎 | | givenName | String | | | |
| 12 | 表示用の名前 (母国語) (*3) | 日本 太郎 | displayName | | String | | | |
| 13 | 所属 (主務・母国語) (*3) | 営業部営業 1 課 | department (*1) | | String | | | |
| 14 | 役職 (主務・母国語) (*3) | 課長 | title | | String | | | |
| 15 | 地域 | ja-JP | locale | | String | | | |
| 16 | メールアドレス | — | emails | | Complex | ○ | | |
| 17 | メールアドレス | taro.nippon@com.example.co.jp | | value | string | | | |
| 18 | 優先フラグ | true | | primary | Boolean | | | |
| 19 | 電話番号 | — | phoneNumbers | | Complex | ○ | | |
| 20 | 値の種別 | work | | type | String | | | |
| 21 | 電話番号 | 03-1234-5678 | | value | string | | | |
| 22 | 優先フラグ | true | | primary | Boolean | | | |

| | | | | | | | | |
|----|--------------|-----------|-----------------|--------------------------|---------|---------|---|--|
| 23 | 有効フラグ | true | active | | Boolean | | | |
| 24 | 多言語表現の名前 | — | localNames (*2) | | Complex | ○ | | |
| 25 | 名前の言語 | ja-JP | | locale | String | | | |
| 26 | 名前 | 日本 太郎 | | formatted | String | | | |
| 27 | 名前 (姓) | 日本 | | familyName | String | | | |
| 28 | 名前 (名) | 太郎 | | givenName | String | | | |
| 29 | 表示用の名前 | 日本 太郎 | | display | String | | | |
| 30 | 優先フラグ | true | | primary | Boolean | | | |
| 31 | 値の種別 | ja-JP | | type | String | | | |
| 32 | 所属組織と役職 | — | | organizationalUnits (*2) | | Complex | ○ | |
| 33 | 所属の言語 | ja-JP | | | locale | String | | |
| 34 | 組織コード or 識別子 | 10010000 | value | | String | | | |
| 35 | 組織名 | 営業 1 課 | name | | String | | | |
| 36 | 表示用の組織名 | 営業部営業 1 課 | display | | String | | | |
| 37 | 役職コード or 識別子 | 5000 | titleValue | | String | | | |
| 38 | 役職名 | 課長 | titleName | | String | | | |
| 39 | 表示用の役職名 | 課長 | titleDisplay | | String | | | |
| 40 | 優先フラグ | true | primary | | Boolean | | | |
| 41 | 値の種別 | ja-JP | type | | String | | | |

- 表中、MV 欄に「○」が示された属性は、Multi-Valued 属性である。
- (*1) SCIM Core Schema で EnterpriseUser として定義されているスキーマを使用する。
- (*2) EIWG 拡張スキーマとして定義して使用する。EIWG 拡張スキーマの完全な定義は付録 B.1.2 に掲載する。
- (*3) グローバル企業などでは name, displayName, department, title にローマ字表記を設定し、母国語表記は localNames, organizationalUnits の多言語表現で対応するという用法もある。

String 型の属性は、少なくとも 255 バイトの文字列を扱えるように実装すべきである。文字コードには UTF-8 を使用する。

SCIM Core Schema では、メールアドレスは emails 属性に複数格納することができる。プロビジョニングに使用するメールアドレスは、emails 属性が 1 つの値だけを持つ場合はその値を、2 つ以上の値を持つ場合は primary 属性が true に設定されたものを使用するように実装すべきである。

電話番号を表す `phoneNumbers` 属性で用いる `type` サブ属性（値の種別）の値として、SCIM Core Schema では "home", "work", "mobile", "fax", "pager", "other" の 6 つの値を標準的な値として定義している。企業での利用を想定する場合、内線番号への対応が必要な場合がある。その場合、内線番号を表す値として "extention" を指定するものとする。

SCIM で定義されている `EnterpriseUser` スキーマには、所属に関連する属性として `division` と `department` の 2 つが用意されている。現状、多くのクラウドサービスでは所属の情報を 1 つの属性として必要としているため、当実装ガイドでは `division` 属性を未使用とし、`department` 属性のみを使用する。

当実装ガイドでは、ユーザーの名前を多言語で表現するための属性として `localNames` を、ユーザーが所属する組織や役職を多言語で表現するための属性として `organizationalUnits` を、それぞれ Multi-Valued 属性として定義する。

これら 2 つの属性では、`locale` サブ属性に言語を表す値を設定することで、クラウドサービスが必要とする言語の値を取り出すことに対応する。`locale` サブ属性に設定する値には、[RFC5646] の形式で、IANA Language Subtag Registry に登録された値を用いる。同一言語で異なる表記がある場合は、`script subtag` の値を含めることに対応する。具体的には、次のような値とする。

| No | 表記 | locale サブ属性に使用する値 |
|----|-----------|-------------------|
| 1 | 漢字表記 | ja-JP |
| 2 | よみがな表記 | ja-Hira-JP |
| 3 | ローマ字・英語表記 | en-US |

よみがな表記は「ひらがな」で値を設定するものとする。クラウドサービスが「カタカナ」で値を必要とする場合は、クラウドサービス側で「ひらがな」で受け取った値を「カタカナ」に変換する。

Multi-Valued 属性の場合に、`type` サブ属性を要求する実装系との互換性を保つため、`type` サブ属性に明確な値の意味の定義がない場合は `locale` サブ属性と同一の値を設定する。

ユーザー作成、更新の処理を行なう場合は、ユーザーデータの属性としてこれら必要属性すべてを含め、値を設定する必要がある。値を設定しなかった場合は、空値が指定されたものとして作成、更新が行なわれる。表中で必須の表示がある属性については、必ず空値ではない値を設定しなければならない。

クラウドサービスの特性として、これらの属性すべてを必要としない場合がある。ユーザーの作成、更新処理において必要としない属性が指定されていた場合は、クラウドサービスはその値

を保管することなく破棄し、属性が指定されていなかったものとして処理を続行しなければならない。

クラウドサービスがこれらの属性すべてを必要としない場合、クラウドサービスが、プロビジョニングに必要としているユーザー属性を利用企業に明示することで、利用企業の ID 管理サーバーは、明示された必要ユーザー属性のみに限定してプロビジョニングを行なってもよい。

クラウドサービスとして、これら以外の属性を必要とする場合がある。その場合は、利用企業との間で個別の取り決めを行ない、属性値を受け取るように実装する必要がある。その方法については、当実装ガイドでは定義しない。

localNames 属性の primary 属性が true である要素の各属性値と、name 属性の属性値は一致していることが望ましい。organizationalUnits 属性の primary 属性が true である要素について、name サブ属性は department 属性と、titleName サブ属性は title 属性と一致していることが望ましい。これらの整合性を保つ責務は、利用企業の ID 管理サーバーにある。

3.3.1.1. 組織情報、役職情報のプロビジョニングを実装する際の考慮点

当実装ガイドでは、組織と役職に関する情報を department 属性と title 属性に格納して扱うようにしている。これは、最小限の実装として、ユーザーが所属する主務組織と主務役職の情報をユーザーの属性としてプロビジョニングすることを可能にすることを意図したためである。

一方、クラウドサービスが企業の組織階層や役職の情報を扱う必要がある場合、組織情報や役職情報のプロビジョニングが必要となる場合がある。

企業での利用を想定した場合、division と department の 2 階層ではなく、より深い階層構造を持つ組織情報の表現や、兼務を含むユーザーの所属状態の表現、各所属でのユーザーの役職情報の表現が必要となることがある。その場合は拡張スキーマを定義して専用の組織リソースや役職リソース、そして拡張されたユーザーリソースの属性を使用することになる。

これらの実装方式の検討、ガイドライン化は今後の課題である。

3.3.2. 特別な意味を持つ属性

当実装ガイドでは、クラウドサービスとクラウド利用企業の認証システムが連携する上で必要となる、特別な意味を持つ属性を定義している。

3.3.2.1. ID 管理サーバー上の識別子 (externalId)

ID 管理サーバー上の識別子 (externalId) には、利用企業の ID 管理サーバー上でユーザーを識別する値を使用する。この識別子は利用企業の ID 管理サーバー内で一意であればよい。

3.3.2.2. サービス利用時のユーザー名 (userName)

クラウドサービスを使用するときのユーザー名を指定する。マルチテナント型のクラウドサービスの場合は、ユーザー名にメールアドレスを、占有型のクラウドサービスの場合は、ユーザー名に従業員番号を使うことが多いだろう。

いずれの場合でも、このユーザー名として何を使うべきかについては、クラウドサービスから指定しなければならない。

その場合、ユーザー名の値は利用企業の ID 管理サーバーがプロビジョニングの過程で決定可能な値、もしくは決定可能な情報から導出できる値を選択しなければならない。

3.3.2.3. 企業 OP でのログイン ID (externalUserName)

クラウドサービスが再認証を要求する場合は利用者の操作性を維持するために、ログイン ID は入力済みの状態とし、パスワードなどクレデンシャル情報の入力だけをユーザーに行なわせることが望ましい。そのため、OpenID Connect の `login_hint` パラメータを用いて、ログイン ID の情報を利用企業の認証サーバーに提供する方式を採る。

クラウドサービス上のログイン ID と利用企業の認証サーバー上のログイン ID はしばしば異なるため、`login_hint` パラメータに渡すべき値を `externalUserName` 属性として事前にクラウドサービスへプロビジョニングしておく。

3.3.2.4. ID トークンの Issuer と Subject (`idTokenClaims.issuer`, `idTokenClaims.subject`)

クラウドサービス利用時の認証を利用企業の認証サーバーで行なう場合、利用企業の認証サーバー上のユーザーとクラウドサービス上のユーザーの紐付けが行なえなければならない。

クラウドサービスと利用企業の認証サーバーのそれぞれで使用するログイン名やユーザー ID は、しばしば異なっていることから、紐付けに使用できる情報が必要となる。

この紐付け用の情報として、認証サーバーが ID トークンに含める `iss` クレームの値と `sub` クレームの値を、それぞれ `idTokenClaims.issuer` 属性と `idTokenClaims.subject` 属性として事前にクラウドサービスへプロビジョニングしておく。

3.3.2.5. 有効フラグ (active)

有効フラグは、ユーザーがクラウドサービスを利用できるか否かを表す。ユーザーの有効化・無効化に連動してクラウドサービスのライセンスの割り当てを有効化・無効化するかどうかについては、クラウドサービスの実装に委ねる。

また、有効フラグが `false` の場合、そのユーザーがクラウドサービス上で表示されるかどうかについては、クラウドサービスの特性に合わせて選択するものとする。

有効フラグの値が `true` から `false` へ変更される場合は、クラウドサービスはそのユーザーの無効化をするとともに、そのユーザーの全セッションを破棄しなければならない。

3.3.3. ユーザー情報の操作

プロビジョニングで行なうユーザー情報の操作には次の3つがある。

1. ユーザーの作成
2. ユーザーの更新
3. ユーザーの削除

3.3.3.1. ユーザーの作成

ユーザーを作成する場合は、SCIM のユーザーリソース作成の手順に従い、HTTP POST リクエストでユーザーの情報を送る。

3.3.3.2. ユーザーの更新

ユーザーの更新を行なう場合は、更新対象のユーザーリソースを指定するために、SCIM サーバー上でのユーザーリソースの識別子 (`id`) が必要となる。

利用企業の ID 管理サーバー上でのユーザーの識別子を `externalId` のフィルター条件に指定して、ユーザーリソースを検索することで、SCIM サーバー上でのユーザーリソースの識別子 (`id`) を求めることができる。

ユーザー情報を更新する場合は、SCIM のユーザーリソース更新の手順に従い、HTTP PUT リクエストを用いたユーザー情報の置換を行なう。

3.3.3.3. ユーザーの削除

ユーザーの削除を行なう場合にも、削除対象のユーザーリソースを指定するために、SCIM サーバー上でのユーザーリソースの識別子 (`id`) が必要となる。そのため、まず [3.3.3.2 節] に示した手順でユーザーリソースの識別子 (`id`) を求める。

ユーザーを削除する場合は、SCIM のユーザーリソース削除の手順に従い、HTTP DELETE リクエストを送る。

3.4. ユーザー認証

利用者がクラウドサービスを利用する場合に、クラウドサービスは利用企業の認証サーバーと連携し、認証処理を行なう。この認証連携には **OpenID Connect** を使用する。認証結果は ID トークンとしてクラウドサービスへ連携される。

3.4.1. ID トークン

ID トークンには、最低限必要な以下のクレームを含める。

| No | クレーム名 | 例 | 説明 |
|----|-----------|------------------------------|---|
| 1 | iss | https://op.com.example.co.jp | Issuer 識別子。利用企業の認証サーバーの URL を指定する。 |
| 2 | sub | e1234567 | Subject 識別子。認証サーバー内でのユーザーの識別子。 |
| 3 | aud | pWBoRam9sG | ID トークンの使用を想定するクライアントの一覧。クラウドサービスの client_id を格納する。 |
| 4 | nonce | q8k-upBX4Z_A | リプレイ攻撃軽減のために用いる ID トークンとクライアントセッションを関連付ける文字列。認証要求に含められた nonce パラメータの値を格納する。 |
| 5 | exp | 1435709162 | ID トークンの有効期限の時刻 |
| 6 | iat | 1435708862 | ID トークンが発行された時刻 |
| 7 | auth_time | 1435708862 | エンドユーザーの認証が行なわれた時刻。後述の再認証要求のみで使用する。 |

sub (Subject 識別子) は、OpenID プロバイダーの実装により決まる。当実装ガイドでは sub の値として、利用企業の認証サーバーで認証を受けるときに使用するログイン ID (従業員番号やメールアドレス) が使用されることを期待している。sub の値は再割り当てができない値であることが要求されているため、独自の値が設定される場合もある。

ID トークンは、署名アルゴリズムに RS256 を用いて、JWS (JSON Web Signature) [RFC7515] で署名されていなければならない。

3.4.2. 認証フロー

利用企業の認証サーバーは、利用企業のセキュリティポリシーにより、インターネットからアクセス可能な場所に設置できるとは限らない。そのため、OpenID Connect による認証フローとして、利用企業の認証サーバー (OP) とクラウドサービス (RP) が直接通信できない場合にも利用できる Implicit フローを使用する。

クラウドサービスへのアクセス方法、認証処理を行なうタイミングにより、次の3つのストーリーへの対応が必要となる。

1. 利用者がクラウドサービスにアクセスし、利用企業の認証サーバーで認証を受けた後に、クラウドサービスを利用する。(以下この利用方法を「クラウドサービス起点のログイン」と呼ぶ。)
2. 利用者が、利用企業の認証サーバーで認証された状態でクラウドサービスにアクセスし、クラウドサービスが利用企業の認証サーバーへ認証状態を確認し、クラウドサービスを利用する。(以下この利用方法を「利用企業起点のログイン」と呼ぶ。)
3. クラウドサービスを使用中に、重要な処理を行なう前に利用者を再度認証するために、利用企業の認証サーバーでの認証を求め、認証ができた場合にのみ処理を継続する。(以下この利用方法を「再認証要求」と呼ぶ。)

認証のために使用する OpenID Connect の各エンドポイントアクセス時の通信路は、TLS による暗号化が行なわれている必要がある。

3.4.2.1. クラウドサービス起点のログイン

この方法は、認証されていないユーザーがクラウドサービスにアクセスした時に、利用企業の認証サーバーで認証を受けた後に、クラウドサービスを利用するケースで使用するログイン方法である。

クラウドサービスが認証を必要とする場合は、OpenID Connect に定められた Implicit フローの手順で利用企業の認証サーバーへ認証を要求する。

利用企業の認証サーバーは、企業が要求する方式でユーザーの認証を行ない、認証結果を ID トークンの形式でクラウドサービスへ返す。

クラウドサービスは、認証要求時に指定したリダイレクト先のエンドポイントで URI フラグメントとして渡される ID トークンを受け取り、OpenID Connect の仕様に定められた手順に従って正当性を検証する。そして [3.4.3 節] に示す方法を用いてクラウドサービス上のユーザーと紐付ける。

認証が完了すれば、最初にユーザーがアクセスしたコンテンツへリダイレクトを行なう。

3.4.2.2. 利用企業起点のログイン

この方法は、例えば、利用企業内でポータルシステムが提供されており、ポータルシステム上のリンクを通じてクラウドサービスへアクセスするなどのケースで使用するログイン方法である。

想定される具体的な利用フローは次のとおりとなる。

1. 利用者は企業のポータル画面へアクセスする。このときに利用企業の認証サーバーで認証済みの状態となる。
2. ポータル画面上のリンクを使って、クラウドサービスのコンテンツへアクセスする。このときにクラウドサービスでの認証が必要となるため、利用企業の認証サーバーへ認証状態を確認する。(認証を要求する)
3. 利用企業の認証サーバーで認証されていることが確認されれば、クラウドサービス上でも認証済みとなり、クラウドサービスのコンテンツにアクセスができる。

この利用フローは、利用企業の認証サーバー上で認証済みの状態で開始されることを除けば、[3.4.2.1 節] に示したクラウドサービス起点のログインと同一であり、クラウドサービス側は同一の実装で対応することができる。

ただし、この場合に利用者に明示的な認証処理を求めることは、利用者が期待した動作と異なる。そのため、利用企業の認証サーバーへ認証要求をする際は `prompt` パラメータを付与してはならない。

3.4.2.3. 再認証要求

この方法は、クラウドサービスを使用中に、重要な処理を行なう前に利用者を再度認証するために、利用企業の認証サーバーでの明示的な認証を求め、認証ができた場合にのみ処理を継続するケースで使用するログイン方法である。

利用企業の認証サーバーへ `OpenID Connect` を用いて再認証を求める場合は、確実に再認証が行なわれるよう、認証要求に次の3つのパラメータを追加する。

1. `prompt=login`
2. `max_age=30`
3. `login_hint`=(プロビジョニング時に `externalUserName` 属性で渡された値)

`prompt` パラメータを指定することで、ユーザーが認証済みでも認証画面を表示することを要求する。

`max_age` パラメータを指定することで、ID トークンに認証処理を行なった時刻の情報が含まれるようになる。

再認証を行なう場合、利用者の操作性を維持するために、ログイン ID は入力済みで、パスワードなどクレデンシャル情報の入力だけとしておくことが望ましい。そのため、`login_hint` パラメータを用いてログイン ID の情報を提供する。

クラウドサービス上のログイン ID と利用企業の認証サーバー上のログイン ID は、しばしば異なっていることがあるため、`login_hint` パラメータに渡す値にはユーザー情報のプロビジョニング時に SCIM の `externalUserName` 属性で渡された文字列を使用しなければならない。

また、再認証要求の場合の ID トークンの検証では、別ユーザーで認証することによる再認証の突破や、代替の ID トークンの挿入による再認証回避を防止するため、ID トークンに含まれる `iss` クレームと `sub` クレームによる同一ユーザーによる再認証であることの検証と、`auth_time` クレームによる認証時刻が妥当な範囲で現在時刻に近いことの検証を行なわなければならない。

3.4.3. ユーザーの紐付け

利用企業の認証サーバーで認証されたユーザーを、クラウドサービス上のユーザーと紐付けることでユーザー認証が完了する。

利用企業の認証サーバーから受け取る ID トークン内の `sub` の値は、クラウドサービス上のユーザー名やユーザー ID と必ずしも一致しない。そのため、ユーザーの紐付けのための仕組みが必要となる。

利用企業の認証サーバー上のユーザーと、クラウドサービス上のユーザーを一意に紐付けるため、クラウドサービスは ID トークンの `iss` クレームと `sub` クレームの値を、それぞれ事前にプロビジョニングされたユーザー属性「ID トークンの Issuer」(`idTokenClaims.issuer`) と「ID トークンの Subject」(`idTokenClaims.subject`) の値と照合しなければならない。

この照合ができなかった場合は、ユーザー認証は失敗したものとしてエラーを返すべきである。

3.5. 強制ログアウト

ユーザーが使用している端末を紛失した場合など、企業では緊急でユーザーのセッションをすべて停止し、システムの利用時に再度認証を求める運用を行なう場合がある。

ユーザーが使用するブラウザを介することなくログアウト処理が行なえることが必要となるため、OpenID Connect Back-Channel Logout 1.0 [OpenID.Backchannel] を使用する。

クラウドサービスの認証サーバーは、どのユーザーのセッションをログアウトするか情報をログアウトトークンに含め、クラウドサービスへ送信することで、ログアウトの要求ができる。

3.5.1. ログアウトトークン

ログアウトトークンは、ID トークンに類似したトークンである。ログアウトトークンには以下のクレームを含める。

| No | クレーム名 | 例 | 説明 |
|----|-------------|------------------------------|---|
| 1 | iss | https://op.com.example.co.jp | Issuer 識別子。利用企業の認証サーバーの URL を指定する。 |
| 2 | sub | e1234567 | Subject 識別子。認証サーバー内でのユーザーの識別子。 |
| 3 | aud | pWBoRam9sG | ログアウトトークンの使用を想定するクライアントの一覧。クラウドサービスの client_id を格納する。 |
| 4 | exp | 1435733336 | ログアウトトークンの有効期限の時刻 |
| 5 | jti | 5ByK | ログアウトトークンの再利用を防止するために付与するこのトークンにユニークに付与される文字列 |
| 6 | logout_only | true | このトークンがログアウトのためのものであることを示す。値は必ず true とする。 |

ログアウトトークンが ID トークンの代わりに利用されることを防ぐため、ログアウトトークンには nonce クレームを含めてはならない。

ログアウトトークンは、署名アルゴリズムに RS256 を用いて、JWS (JSON Web Signature) [RFC7515] で署名されていなければならない。署名には ID トークンの署名に使用する鍵と同じ鍵を用いる。

3.5.2. 強制ログアウト処理

利用企業の認証サーバーは、クラウドサービスが提供するログアウトエンドポイントへログアウトトークンを POST することで、強制ログアウトを要求することができる。

クラウドサービスは、受け取ったログアウトトークンを、OpenID Connect Back-Channel Logout の仕様に定められた手順に従って正当性を検証する。その後、[3.4.3 節] に示した方法でログアウトさせるクラウドサービス上のユーザーを特定し、そのユーザーのセッションをログアウトさせる。

3.6. 鍵・パスワードのメンテナンス

3.6.1. ID トークン署名用キーペアの更新

利用企業の認証サーバーでは ID トークン署名用の秘密鍵を厳重に管理する必要がある。秘密鍵が漏えいした場合、第三者が偽の ID トークンを発行することにより、クラウドサービス上の全アカウントに成りすましアクセスができるようになる。

署名用のキーペアは適切なタイミングでローテーションを行なう必要がある。たとえば、秘密鍵の漏えいが疑われるときや、使用している鍵長が安全ではなくなったときなどに、キーペアのローテーションが必要である。

3.6.2. SCIM エンドポイントアクセス用パスワードの更新

利用企業の ID 管理システムでは、SCIM エンドポイントアクセス用のパスワードを厳重に管理する必要がある。パスワードが漏えいした場合、第三者がクラウドサービス上のユーザー情報を不正に読み出したり、不正に書き換えたりすることができるようになる。

当実装ガイドでは、SCIM エンドポイントアクセス時の認証方式として、少なくとも Basic 認証に対応することを求めている。

Basic 認証で用いるパスワードは、自動的にかつ定期的に交換されるものではなく、人の手を介して交換・設定されるものであるため、設定・運用担当者を通じた漏えいに特に注意する必要がある。

SCIM エンドポイントアクセス用パスワードは、パスワード漏えいが疑われる場合に更新が必要である。

3.6.3. SCIM エンドポイントアクセス時の OAuth を使った認可への対応

SCIM エンドポイントへのアクセスを保護するためには、エンドポイントアクセスのためのクレデンシャル情報（パスワードやトークン）を、人の手を介在せずに交換する方式を用いることが望ましい。このような方式として、OAuth 2.0 による認可と、それによって発行されたトークンを利用する方式がある。

SCIM クライアントとなる利用企業の ID 管理サーバーは、一般的には ID 管理の製品を用いて実装されることが多いものの、OAuth 2.0 の認可フローに対応したものが少ないという現状がある。そこで、当実装ガイドでは、利用企業の認証システムとクラウドサービスの相互接続の実現を早期実現するため、SCIM エンドポイントアクセス時の認証方式として Basic 認証に対応を求めることを選択した。

しかし一方で、Basic 認証で実装する場合は、すでに [3.6.2 節] でも述べたとおり、パスワードの交換・設定が人の手を介して行なわれるため、パスワード漏えいのリスクが残存する。

そのため、SCIM エンドポイントアクセス時の OAuth 2.0 を使った認可にも対応することを強く推奨する。

OAuth 2.0 を使った認可を実装する場合は、次のような方法を用いる。

1. 利用企業の ID 管理サーバーを、クラウドサービスの（OAuth 2.0 が意味するところの）クライアントとして登録する。
2. 利用企業の ID 管理サーバーで SCIM クライアントの構成を行なう際に、OAuth 2.0 の認可フローを用いて、利用企業の管理者であることの認証、SCIM エンドポイントへのアクセス認可の取得、SCIM エンドポイントアクセスのためのトークンの発行を受ける。

3. SCIM エンドポイントアクセス時は、発行されたアクセストークンを使用する。
4. アクセストークンの定期的な交換を実現するために、リフレッシュトークンを用いたアクセストークンの更新に対応する。

この方法のメリットは、大きく 2 つある。ひとつは、利用企業の管理者であることの認証を、多要素認証などを併用したより安全な認証方式で認証できる点である。もうひとつは、SCIM エンドポイントにアクセスするためのトークンが、人の手を介在することなく、定期的にかつ自動的に交換される運用を実現できる点である。

第4章 クラウドサービス事業者向け実装ガイド

4.1. OpenID Connect RP（クライアント）の実装

この節では、クラウドサービスが利用企業の認証サーバーに認証を要求し、認証結果を受け取る OpenID Connect リライディング・パーティとしての実装について解説する。

4.1.1. 認証フローの概要

認証フローは OpenID Connect の Implicit フローを用いる。実装する認証フローの全体を以下に示す。

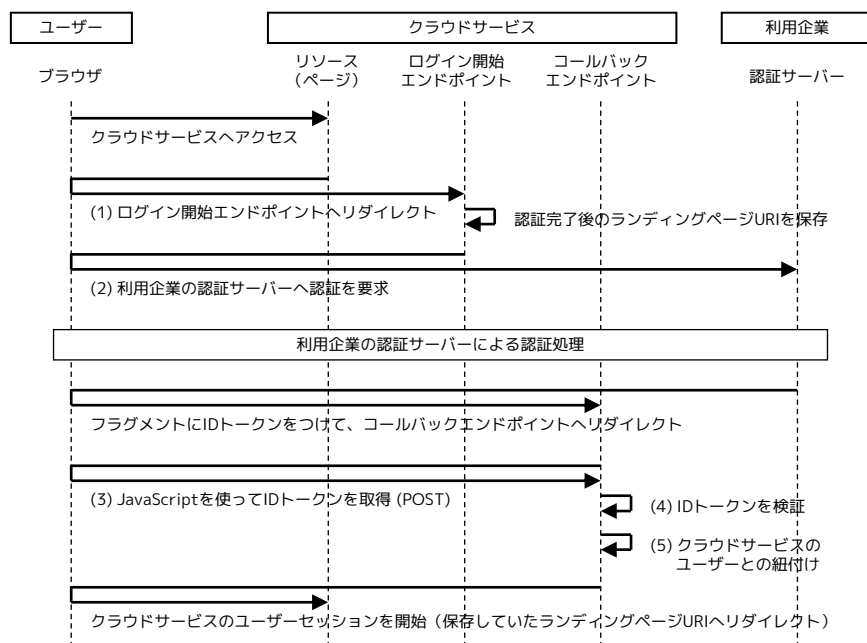


図 4-1: OpenID Connect Implicit フローの RP 処理シーケンス

4.1.2. ログイン開始エンドポイントの実装

認証されていないユーザーがクラウドサービスにアクセスした際は、クラウドサービスは利用企業の認証サーバーへ認証を要求し、認証が完了すれば、最初にユーザーがリクエストしたページ（以下ランディングページと呼ぶ）を表示する。

そのため、クラウドサービスは利用企業の認証サーバーへ認証要求をする前に、このランディングページの URI を保管する機能を実装する必要がある。この実装方法はクラウドサービスが自サービスに適した方式に決めればよい。

以下では、実装の一例として、OpenID Connect Core 1.0 [OpenID.Core] の Section.4 で定められている "login initiation endpoint" に沿った実装について解説する。

ログイン開始エンドポイントでは、次の3つのパラメータを受け取る。

| No | パラメータ | 説明 |
|----|-----------------|---|
| 1 | iss | 利用企業の認証サーバーの URL (必須パラメータ) |
| 2 | login_hint | 利用企業の認証サーバーでのログイン ID を指定する。再認証要求の場合に必要。 |
| 3 | target_link_uri | ランディングページの URI を指定する。 |

オープンリダイレクタとして不正に利用されることを予防するため、クラウドサービスは、target_link_uri として、クラウドサービス内の URI が指定されていることを検証しなければならない。target_link_uri が指定されていない場合のランディング先は、クラウドサービスが適切に決定する。

ログイン開始エンドポイントでは、ログイン処理専用のセッションを開始し、target_link_uri に指定された値と、以降の認証要求で使用する state パラメータを紐付けて保存する。

ログイン開始エンドポイントへのリダイレクトの例を以下に示す。

| No | 情報 | 例 |
|----|----------------|---|
| 1 | ログイン開始エンドポイント | https://www.svc.example.net/auth |
| 2 | ユーザーがアクセスしたページ | https://www.svc.example.net/c765421/top |
| 3 | 利用企業の認証サーバー | https://op.com.example.co.jp |

```
HTTP/1.1 302 Found
Location: https://www.svc.example.net/auth?
  iss=https%3A%2F%2Fop.com.example.co.jp
  &target_link_uri=https%3A%2F%2Fwww.svc.example.net%2Fc7654321%2Ftop
```

4.1.3. 利用企業の認証サーバーへの認証要求

4.1.3.1. 通常の認証要求

通常の認証要求では、利用企業の認証サーバーに対して次のパラメータを含める。

| No | パラメータ | 例 | 説明 |
|----|---------------|--|--|
| 1 | response_type | id_token | Implicit フローで ID トークンを受け取るため "id_token" を指定する |
| 2 | client_id | pWBoRam9sG | 利用企業の認証サーバーがクラウドサービスに割り当てた ID |
| 3 | redirect_uri | https%3A%2F%2Fwww.svc.example.net%2Fcb | 認証後のリダイレクト先 URI。フラグメントに ID トークンがついた状態でリダイレクトされる。 |
| 4 | scope | openid | OpenID Connect を使用するため "openid" を指定する |
| 5 | state | k4y97klszxi | 認証要求と認証後のリダイレクトアクセスを関連付ける文字列 |
| 6 | nonce | q8k-upBX4Z_A | クライアントセッションと ID トークンを関連付ける文字列 |

state パラメータは、認証要求と認証後に受け取る redirect_uri で行なわれるリダイレクトアクセスを関連付け、CSRF 攻撃を防止する値として使用する。認証要求毎に暗号論的に安全なランダムな値を生成して使用する。

nonce パラメータは、クライアント（ブラウザ）セッションと ID トークンを関連付け、リプレイ攻撃（ID トークンの使い回し）を防止する値として使用する。クライアントセッション毎に暗号論的に安全なランダムな値を生成し、クライアントに保管して使用する。OpenID Connect Core では、この実装方法として、ブラウザのセッション Cookie に HttpOnly で保存する方法が記載されている。（OpenID Connect Core 1.0 [OpenID.Core] の Section 15.5.2）

認証要求は、認証サーバーへのリダイレクトによって行なう。認証要求の例を以下に示す。

```
HTTP/1.1 302 Found
Location: https://op.com.example.co.jp/authorize?
  response_type=id_token
  &client_id=pWBoRam9sG
  &redirect_uri=https%3A%2F%2Fwww.svc.example.net%2Fcb
  &scope=openid
  &state=k4y97klszxi
  &nonce=q8k-upBX4Z_A
```

4.1.3.2. 再認証のための認証要求

再認証のための認証要求では、通常の認証要求に加え、次のパラメータを含める。

| No | パラメータ | 例 | 説明 |
|----|------------|----------|--|
| 1 | prompt | login | 認証済みであっても認証画面を必ず表示する。"login" を指定する。 |
| 2 | max_age | 30 | ID トークンに auth_time クレームを含める。30 秒を指定する。 |
| 3 | login_hint | e1234567 | 企業 OP でのログイン ID (externalUserName) としてプロビジョニングされた値 |

max_age パラメータはその意味合いから値が 0 であることが望ましいが、max_age の値が小さすぎると正しく動作しない実装があるため、0 ではなく小さめの整数値を時間として設定する。

認証要求の例を以下に示す。

```
HTTP/1.1 302 Found
Location: https://op.com.example.co.jp/authorize?
  response_type=id_token
  &prompt=login
  &max_age=30
  &login_hint=e1234567
  &client_id=pWBoRam9sG
  &redirect_uri=https%3A%2F%2Fwww.svc.example.net%2Fcb
  &scope=openid
  &state=k4y97k1szxi
  &nonce=q8k-upBX4Z_A
```

4.1.4. 利用企業の認証サーバーからの認証結果の受け取り

利用企業の認証サーバーでの認証結果は、認証要求の際に指定した redirect_uri にフラグメントとして付加された URI へリダイレクトされることでクラウドサービスへ渡される。

利用企業の認証サーバーからブラウザへ返される応答の例を以下に示す。

```
HTTP/1.1 302 Found
Location: https://www.svc.example.net/cb#
  id_token=eyJhbGciOiAi... .eyJpc3MiOiAi... .rZR3EjzaHA ...
  &state=k4y97k1szxi
```

この結果、クラウドサービスは次のリクエストを受け取る。

```
GET /cb HTTP/1.1
Host: www.svc.example.net
```

フラグメントに付与された ID トークンを受け取るためには、フラグメントの値を HTTP POST する JavaScript を実行する必要がある。そのため、GET による `redirect_uri` へのアクセスでは JavaScript を応答する。

JavaScript の実装については標準としての取り決めはないため、クラウドサービスが自サービスに適した方式で実装すればよい。OpenID Connect Core では実装方法の例として以下のコードが紹介されている。(OpenID Connect Core 1.0 [OpenID.Core] の Section 15.5.3)

この例では、"`https://www.svc.example.net/catch_response`" へ POST し、POST が成功すれば、"`https://www.svc.example.net/redirect_after_login`" へリダイレクトするようになっている。

```
HTTP/1.1 200 OK
Content-Type: text/html

<script type="text/javascript">

// First, parse the query string
var params = {}, postBody = location.hash.substring(1),
    regex = /(?:^&=+)=([^\&]*)/g, m;
while (m = regex.exec(postBody)) {
    params[decodeURIComponent(m[1])] = decodeURIComponent(m[2]);
}

// And send the token over to the server
var req = new XMLHttpRequest();
// using POST so query isn't logged
req.open('POST', 'https://' + window.location.host +
    '/catch_response', true);
req.setRequestHeader('Content-Type',
    'application/x-www-form-urlencoded');

req.onreadystatechange = function (e) {
    if (req.readyState == 4) {
        if (req.status == 200) {
            // If the response from the POST is 200 OK, perform a redirect
            window.location = 'https://'
                + window.location.host + '/redirect_after_login'
        }
        // if the OAuth response is invalid, generate an error message
        else if (req.status == 400) {
            alert('There was an error processing the token')
        } else {
            alert('Something other than 200 was returned')
        }
    }
};
req.send(postBody);
```

4.1.4.1. 認証成功時の応答

認証に成功した場合、フラグメントで2つのパラメータを受け取る。

| No | パラメータ | 例 | 説明 |
|----|----------|---|---------------------|
| 1 | id_token | eyJhbGciOieyJpc3MiOirZR3EjzaHA... | 発行された ID トークン |
| 2 | state | k4y97klszxi | 認証要求時に指定された state 値 |

[4.1.4 節] で例示した JavaScript を使用した場合は、以下のようなリクエストとして認証結果を受け取ることになる。

```
POST /catch_response HTTP/1.1
Host: www.svc.example.net
Content-Type: application/x-www-form-urlencoded

id_token=eyJhbGciOi ... .eyJpc3MiOi ... .rZR3EjzaHA ...&state=k4y97klszxi
```

認証結果を受け取ると、最初に state 値が認証要求時に指定した値と同一であるかどうかを検証する。

state 値が一致しない場合は認証エラーとし、ユーザーを適切なページへ誘導する必要がある。適切なページとは、エラーメッセージを表示するページのことを指す。

state 値が一致した場合は、ID トークンの検証へ進む。

4.1.4.2. 認証失敗時の応答

認証に失敗した場合、フラグメントで2つのパラメータを受け取る。

| No | パラメータ | 例 | 説明 |
|----|-------------------|-----------------------------|--|
| 1 | error | access_denied | エラーコード (OAuth 2.0 [RFC6749] の Section 4.1.2.1, OpenID Connect Core 1.0 [OpenID.Core] の Section 18.3.1 のコードを取りうる) |
| 2 | error_description | User authentication failed. | エラーの詳細 (このパラメータはオプションである) |
| 3 | state | k4y97klszxi | 認証要求時に指定された state 値 |

[4.1.4 節] で例示した JavaScript を使用した場合は、以下のようなリクエストとして認証結果を受け取ることになる。

```
POST /catch_response HTTP/1.1
Host: www.svc.example.net
Content-Type: application/x-www-form-urlencoded

error=access_denied&error_description=User%20authentication%20failed.&state=k4y97k1szxi
```

認証結果を受け取ると、最初に `state` 値が認証要求時に指定した値と同一であるかどうかを検証する。

`state` 値が一致しない場合は、受け取ったレスポンス（エラーコードとエラーの詳細）は信頼することができない。

認証失敗の応答を受け取った場合は、`state` 値の検証結果にかかわらず認証エラーとし、ユーザーを適切なページへ誘導する必要がある。適切なページとは、エラーメッセージを表示するページのことを指す。

4.1.5. ID トークンの検証

4.1.5.1. 通常の認証要求の場合の ID トークンの検証

ID トークンは、ここまで解説した手順を使用した場合、JWS (JSON Web Signature) [RFC7515] で署名された JWT (JSON Web Token) [RFC7519] の形式で受け取ることになる。

受け取る ID トークンの例を以下に示す。base64url エンコーディングされた、`!` (ピリオド) で区切られた 3 つの部分から構成される。

```
eyJhbGciOiJIUzI1NiIsImtpZCI6Im1BdzUifQ.eyJpc3MiOiJodHRwczovL29wLmNvbS5leGFtcGx1LmNvLmpwIiwic3ViIjoizTEyMzQ1NjciLCJhdWQiOiJwV0JvUmFtOXNHIiwibm9uY2UiOiJxOGstdXBWDRaX0EiLCJleHAiOjE0MzU3MDkxNjIsIm1hdCI6MTQzNTcwODg2Mn0.rZR3EjzaHA_BCXiIkD0KctiFYnRd7hZriG36-Yx7iEckVPG2srQ0nNIP9u5KHIG6qzhHu2PBR8tjvYu2a1dL8pf-VNAhD3AMYYju_TQ2qXxp5o08VuqX7rc-vV4hjXcwPR8CUsXveJwpLpI1TPcZyMIDkBVZNCWPRDgtAJwDYtLmFX-ONf1Y9sMvwdVBBaNFxRiCXPwb5bhtDGM717KW1rwnrVo11av6KZUuGAXuwUqQt085GftsfgMbh3cj9W5xEH9QOKULy4C2r6qKIydCz99WkUnoU1fDz50mjSdLgJnfCo1xYZjvJnyoAu3BUu0knE7sNpzDwn4DxJLVhHg
```

1 番目のパートは JOSE ヘッダになっている。少なくとも次の 2 つの属性が含まれている。

| No | 属性 | 例 | 説明 |
|----|-----|-------|--------------------------------------|
| 1 | alg | RS256 | 署名アルゴリズム。当実装ガイドでは RS256 への対応を指定している。 |
| 2 | kid | iAw5 | 署名検証用の公開鍵の ID |

JOSE ヘッダの例を以下に示す。

```
{
  "alg": "RS256",
  "kid": "iAw5"
}
```

2 番目のパートは、ID トークンのクレームとなっている。少なくとも [3.4.1 節] に示したクレームが含まれている。

ID トークンのクレームの例を以下に示す。通常の認証要求の場合、`auth_time` は含まれない。(含まれていても良い)

```
{
  "iss": "https://op.com.example.co.jp",
  "sub": "e1234567",
  "aud": "pWBoRam9sG",
  "nonce": "q8k-upBX4Z_A",
  "exp": 1435709162,
  "iat": 1435708862
}
```

ID トークンの検証として、最低限、次の事項を検証する。

1. `iss` クレームが、認証要求を送った利用企業の認証サーバーの URL と一致することを確認する。
2. `aud` クレームが、自クラウドサービスに割り当てられた `client_id` であることを検証する。

もし `aud` クレームが複数の値を持つ場合は、OpenID Connect Core 1.0 [OpenID.Core] の Section 3.1.3.7 の定義に従った追加の検証が必要である。

3. ID トークンの署名を、RS256 アルゴリズムを用いて検証する。

ID トークンの `base64url` エンコーディングされた 1 番目、2 番目のパートを `!` で連結した文字列を、ハッシュ関数に SHA-256 を使うように構成された RSASSA-PKCS1-v1_5 署名検証器にかけることで検証を行なう。

具体的な手順は JWS [RFC7515] の Section 5.2, Section A.2 などに示されている。署名検証に用いる公開鍵の取得方法は [4.1.9 節] に示す。

もし、ID トークンに RS256 以外の署名アルゴリズムが指定されており、その署名をクラウドサービスが検証可能な場合は、その検証で代替しても良い。

4. 現在時刻が `exp` クレームで指定された時刻よりも前であることを検証する。
5. `nonce` クレームが、クライアント (ブラウザ) セッションと紐付いた `nonce` 値と一致することを確認する。

6. logout_only クレームが含まれていないことを検証する。

ID トークンの検証に失敗した場合は認証エラーとし、ユーザーを適切なページへ誘導する必要がある。適切なページとは、エラーメッセージを表示するページのことを指す。

ID トークンの検証に成功した場合は、認証されたユーザーとクラウドサービスのユーザーとの紐付けに進む。

4.1.5.2. 再認証要求の場合の ID トークンの検証

再認証要求を行なった場合には、ID トークンのクレームに auth_time が必ず含まれる。

```
{
  "iss": "https://op.com.example.co.jp",
  "sub": "e1234567",
  "aud": "pWBoRam9sG",
  "nonce": "q8k-upBX4Z_A",
  "exp": 1435710062,
  "iat": 1435709762,
  "auth_time": 1435709762
}
```

再認証要求を行なった場合の ID トークンの検証では、別ユーザーでの再認証の突破や代替の ID トークンの挿入による再認証回避を防止するため、[4.1.5.1 節] に示した ID トークンの検証事項に加え、次の事項を検証する。

1. iss クレームと sub クレームの値がそれぞれ、現在サービスを使用しているユーザーの「ID トークンの Issuer」(idTokenClaims.issuer) と「ID トークンの Subject」(idTokenClaims.subject) の値と一致していることを検証する。
2. auth_time クレームの値が、妥当な範囲で現在時刻に近いことを検証する。

ID トークンの検証に失敗した場合は認証エラーとし、ユーザーを適切なページへ誘導する必要がある。適切なページとは、エラーメッセージを表示するページのことを指す。

ユーザーの操作性を考慮すると、ID トークンの検証に失敗した場合でも再認証要求前のセッション状態を維持し、再認証を必要とする処理へのリトライができるようにサービスを実装することが必要である。

ID トークンの検証に成功した場合は再認証が成功したものとみなし、ユーザーが要求した処理を実行する。

4.1.6. 認証されたユーザーとクラウドサービスのユーザーとの紐付け

利用企業の認証サーバーで認証されたユーザーは、ID トークンの `iss` クレームと `sub` クレームの値の組み合わせで識別される。

このユーザーをクラウドサービスのユーザーに紐付けるために、事前にプロビジョニングされたユーザー属性「ID トークンの Issuer」(`idTokenClaims.issuer`) と「ID トークンの Subject」(`idTokenClaims.subject`) の値との照合を行なう。

この照合に失敗した場合は認証エラーとし、ユーザーを適切なページへ誘導する必要がある。適切なページとは、エラーメッセージを表示するページのことを指す。

照合に成功した場合はクラウドサービスのユーザーセッションを開始し、認証前に保存したランディングページへリダイレクトする。

4.1.7. セッションの管理

クラウドサービスのユーザーセッションの管理は、クラウドサービスの実装に委ねるものとする。

4.1.8. 強制ログアウト処理

利用企業で利用者のセッションをすべて停止する必要が発生した場合、利用企業の認証サーバーは OpenID Connect Back-Channel Logout 1.0 [`OpenID.Backchannel`] を用いて、ログアウト要求を送ってくる。

4.1.8.1. 利用企業の認証サーバーからのログアウト要求の受け取り

利用企業の認証サーバーからログアウト要求を受け取るため、ログアウトエンドポイントを実装する。ログアウトエンドポイントでは次のパラメータを HTTP POST を用いて受け取る。

| No | パラメータ | 例 | 説明 |
|----|---------------------------|---|-----------|
| 1 | <code>logout_token</code> | <code>eyJhbGciOiAi... .eyJpc3MiOiAi... .zPt-EH3b9m ...</code> | ログアウトトークン |

`logout_token` パラメータが含まれていない場合は、[4.1.8.5 節] に示す方法でエラーを応答する。

4.1.8.2. ログアウトトークンの検証

ログアウトトークンは、JWS (JSON WEb Signature) で署名された JWT (JSON Web Token) [RFC7519] の形式で受け取る。

ログアウトトークンの検証の方法は、ID トークンの検証方法 [4.1.5.1 節] に準じた方法となる。ログアウトトークンでは、最低限、次の事項を検証する。

1. iss クレームが、クラウドサービスに登録された利用企業の認証サーバーの URL と一致することを検証する。
2. aud クレームが、自クラウドサービスに割り当てられた client_id であることを検証する。
3. ログアウトトークンの署名を、RS256 アルゴリズムを用いて検証する。
4. 現在時刻が exp クレームで指定された時刻よりも前であることを検証する。
5. logout_only クレームの値が true であることを検証する。
6. nonce クレームが含まれていないことを検証する。
7. jti クレームの値が同一の別のログアウトトークンを、このリクエストの少し前の時間に受け取っていないことを検証する。この検証はオプションである。

ログアウトトークンの検証に失敗した場合は、[4.1.8.5 節] に示す方法でエラーを応答する。

4.1.8.3. ログアウト対象のユーザーとクラウドサービスのユーザーとの紐付け

ログアウト対象のユーザーは、ログアウトトークンの sub クレームと iss クレームを用いて識別される。認証のときと同じ方法 [4.1.6 節] で、このユーザーをクラウドサービスのユーザーに紐付ける。

紐付けに成功すれば、クラウドサービスに適した方法を用いて、そのユーザーのすべてのセッションについてログアウト処理を行なう。

ログアウト処理中にエラーが発生した場合は、[4.1.8.5 節] に示す方法でエラーを応答する。

ログアウト対象ユーザーの有効なセッションが存在しなかった場合でも、ログアウト処理が正しく行なえたものとして振舞う必要がある。

4.1.8.4. ログアウト処理成功時の応答

ログアウト処理が成功した場合は、HTTP ステータスコード 200 (OK) を応答する。このとき、レスポンスがキャッシュされないように、キャッシュ制御用の HTTP レスポンスヘッダをつける必要がある。

```
HTTP/1.1 200 OK
Cache-Control: no-store
Pragma: no-cache
```

4.1.8.5. ログアウト処理失敗時の応答

ログアウト処理が失敗した場合は、その内容に応じて次のような HTTP ステータスコードを応答する。

| No | エラーの内容 | ステータスコード |
|----|--|----------|
| 1 | リクエストにエラーがある。 | 400 |
| 2 | ログアウトトークンの検証に失敗した。 | 400 |
| 3 | クラウドサービスのユーザーとの紐付けに失敗した。 | 400 |
| 4 | クラウドサービスでのログアウト処理が失敗した。 | 501 |
| 5 | 関連サイトのログアウトを併せて行なう必要があるが、そのログアウト処理に失敗した。 | 504 |

HTTP/1.1 400 Bad Request

強制ログアウト処理を行なう場合の OpenID Connect Back-Channel Logout のリクエスト・レスポンスの例を [付録 A.5] に掲載する。

4.1.9. 利用企業の認証サーバーの公開鍵の取得

クラウドサービスが署名検証用の公開鍵を受け取る方法として、2つの方法がある。

1. クラウドサービスが提供する利用企業の管理者向け機能を使い、認証サーバーの登録画面から署名検証用の公開鍵を登録してもらう。
2. 利用企業が署名用公開鍵を提供するエンドポイントを実装し、その URI をクラウドサービスへ `jwtks_uri` として設定し、そのエンドポイントから公開鍵を受け取る。

利用企業が `jwtks_uri` を公開しない場合に対応するために、クラウドサービスは少なくとも方法 1 に対応しなければならない。

利用企業が `jwtks_uri` を公開している場合は、方法 2 の `jwtks_uri` から公開鍵を受け取る方式を用いても良い。その場合、利用企業の担当者が使用しない公開鍵を登録したままにすることによる不正な ID トークンの受け入れリスクを回避するため、方法 1 との併用はしてはならない。(方法 2 だけを用いなければならない)

方法 1、方法 2 のいずれの方法を用いる場合でも、利用企業が公開鍵のローテーションを行なうことを想定し、複数の公開鍵を扱えるようにしておくことを推奨する。方法 2 を用いる場合の公開鍵ローテーションに合わせた `jwtks_uri` からの公開鍵の取り込みには、OpenID Connect Core 1.0 [OpenID.Core] の Section 10.1.1 に示された方法を用いる。

ID トークンの署名検証を行なう場合は、JOSE ヘッダに kid 属性として指定された ID を持つ公開鍵を用いなければならない。方法 1 を用いる場合には、公開鍵の登録時に同じ ID を持つ二つ以上の公開鍵の登録を受け付けてはならない。方法 2 を用いる場合、同一の ID を持つ複数の公開鍵が JWK Set に含まれているときの処理は、JWK (JSON Web Key) [RFC7517] の Section 5 に示された方法を採用すべきである。

4.2. SCIM サーバーの実装

この節では、クラウドサービスがプロビジョニング要求を受け付ける SCIM サーバーの実装について解説する。

4.2.1. User エンドポイントの実装

SCIM サーバーは、ユーザー情報のプロビジョニング処理で使用する User エンドポイントを実装していなければならない。以降の解説では、エンドポイントの URI が以下のとおりであるものとしている。

| No | エンドポイント | URI |
|----|-------------------|--|
| 1 | SCIM User エンドポイント | https://scim.svc.example.net/User https://scim.svc.example.net/v2/User |

今後の SCIM プロトコルのバージョンの変更により、エンドポイントの変更が必要となる場合がある。そのため、プロトコルのバージョンを URI に含める SCIM Protocol Versioning に対応することが望ましい。

このエンドポイントでは、POST, PUT, DELETE の 3 つの操作に対応する。

4.2.1.1. ユーザーの作成 (POST)

4.2.1.1.1. リクエストの受け取り

ユーザーの作成は、HTTP POST メソッドでユーザーデータを受け取ることで実行される。SCIM クライアントから送られてくるリクエストの例を以下に示す。

```
POST /Users HTTP/1.1
Host: scim.svc.example.net
Accept: application/scim+json
Content-Type: application/scim+json
Authorization: Basic Yzc2NTQzMjE6Z3l0OUUpSN3pUWDRteXd0NHFiMW5ZdUhpTTA3WF10Q1U=
Content-Length: ...

{
  "schemas": [
    "urn:ietf:params:scim:schemas:core:2.0:User",
    "urn:ietf:params:scim:schemas:extension:enterprise:2.0:User",
    "urn:oidfj:params:scim:schemas:extention:enterprisejp:2.0:User"
  ],
  "userName": "taro.nippon@com.example.co.jp",
  "externalId": "e1234567",
  ... (ユーザーデータ。以下省略) ...
}
```

リクエストで受け取ったユーザーデータは、[4.2.3 節] に示す手順で検証を行なう。

4.2.1.1.2. クラウドサービスのユーザーとして登録

受け取ったユーザーデータの検証ができた場合は、ユーザーデータをクラウドサービスのユーザーとして登録する。

ユーザーデータとして渡されたデータの中に、クラウドサービスで使用しない属性が含まれていた場合、その情報は保管することなく破棄しなければならない。

ユーザーデータとして渡されたデータが [3.3.1 節] に示した属性を含んでいなかった場合は、クラウドサービスが適切な属性値（空でも良い）を割り当てる。

ユーザーデータに含まれる属性「ID トークンの Issuer」(idTokenClaims.issuer)、「ID トークンの Subject」(idTokenClaims.subject) と「企業 OP でのログイン ID」(externalUserName) については、認証処理で使用する属性となるため、クラウドサービスのユーザーの属性として保管する必要がある。

SCIM サーバーは、クラウドサービスのユーザー属性として、次の 4 つの管理用属性を付与して保管する必要がある。

1. ID

ユーザーリソースを SCIM サーバー（クラウドサービス）内で一意に識別するためのユーザーリソース識別子を生成して設定する。この値は、クラウドサービスを使用する全利用企業の全ユーザーの中で一意である必要がある。この値はユーザーリソースの "id" 属性として用いる。

2. バージョン

SCIM サーバーで、HTTP ETag を用いた Resource Versioning に対応する場合に、バージョンをあらわす値を生成して設定する。この値はユーザーリソースの "meta.version" 属性として用いる。この値はオプションである。

3. 作成日

このユーザーリソースを作成した日時を設定する。この値はユーザーリソースの "meta.created" 属性として用いる。

4. 更新日

作成日と同じ値を設定する。この値はユーザーリソースの "meta.lastModified" 属性として用いる。

付与される属性値の例を以下に示す。

| No | 属性 | 値 |
|----|-------|--------------------------------------|
| 1 | ID | a2e492da-e2ed-4d90-a186-6fc01b56b8d9 |
| 2 | バージョン | 4124bc0a9335c27f086f24ba207a4912 |
| 3 | 作成日 | 2015-04-01T01:23:45.678Z |
| 4 | 更新日 | 2015-04-01T01:23:45.678Z |

4.2.1.1.3. 登録結果の応答

ユーザーの登録が成功した場合は、HTTP ステータスコード 201 (Created) とともに、登録されたユーザーリソースを応答する。

```
HTTP/1.1 201 Created
Content-Type: application/scim+json
Location: https://scim.svc.example.net/v2/User/a2e492da-e2ed-4d90-a186-6fc01b56b8d9
ETag: W/"4124bc0a9335c27f086f24ba207a4912"

{
  "schemas": [
    "urn:ietf:params:scim:schemas:core:2.0:User",
    "urn:ietf:params:scim:schemas:extension:enterprise:2.0:User",
    "urn:oidfj:params:scim:schemas:extention:enterprisejp:2.0:User"
  ],
  "id": "a2e492da-e2ed-4d90-a186-6fc01b56b8d9",
  "userName": "taro.nippon@com.example.co.jp",
  "externalId": "e1234567",
  "meta": {
    "resourceType": "User",
    "created": "2015-04-01T01:23:45.678Z",
    "lastModified": "2015-04-01T01:23:45.678Z",
    "location": "https://scim.svc.example.net/v2/User/a2e492da-e2ed-4d90-a186-6fc01b56b8d9",
    "version": "W/\\"4124bc0a9335c27f086f24ba207a4912\""
  },
  ... (ユーザーデータ。以下省略) ...
}
```

クラウドサービス内の処理が理由でユーザーの登録中にエラーが発生した場合は、HTTP ステータスコード 500 (Internal Server Error) を応答する。このとき、利用企業の ID 管理サーバーにエラーの内容を伝達するため、detail にエラーの詳細を記述して応答することが望ましい。

```
HTTP/1.1 500 Internal Server Error
Content-Type: application/scim+json

{
  "schemas": ["urn:ietf:params:scim:api:messages:2.0:Error"],
  "detail": "SCIM サーバーがエラーの詳細を記入する",
  "status": "500"
}
```

SCIM を用いてユーザーを作成する場合の SCIM リクエスト・レスポンスの例を [付録 B.2] に掲載する。

4.2.1.2. ユーザーの更新 (PUT)

4.2.1.2.1. リクエストの受け取り

ユーザーの更新は、HTTP PUT メソッドでユーザーデータを受け取ることで実行される。SCIM クライアントから送られてくるリクエストの例を以下に示す。

HTTP PUT によるユーザー情報の置き換えのため、SCIM クライアントは変更の有無にかかわらず、ユーザーデータの全属性を送ってくる。

```
PUT /Users/a2e492da-e2ed-4d90-a186-6fc01b56b8d9 HTTP/1.1
Host: scim.svc.example.net
Accept: application/scim+json
Content-Type: application/scim+json
Authorization: Basic Yzc2NTQzMjE6Z3l0OUUpSN3pUWDRteXd0NHFiMW5ZdUhpTTA3WF10Q1U=
Content-Length: ...
If-Match: W/"4124bc0a9335c27f086f24ba207a4912"

{
  "schemas": [
    "urn:ietf:params:scim:schemas:core:2.0:User",
    "urn:ietf:params:scim:schemas:extension:enterprise:2.0:User",
    "urn:oidfj:params:scim:schemas:extention:enterprisejp:2.0:User"
  ],
  "id": "a2e492da-e2ed-4d90-a186-6fc01b56b8d9",
  "userName": "taro.nippon@com.example.co.jp",
  "externalId": "e1234567",
  ... (更新内容を含むユーザーデータ。以下省略) ...
}
```

URI に指定されたユーザーリソースが見つからない場合は、HTTP ステータスコード 404 (Not Found) を応答する。

リクエストで受け取ったユーザーデータは、[4.2.3 節] に示す手順で検証を行なう。

HTTP ETag を用いた Resource Versioning に対応している場合には、If-Match ヘッダに指定されたバージョンと、クラウドサービスで管理しているユーザー情報のバージョンが一致していることを確認する。もしバージョンが異なっている場合は、HTTP ステータスコード 409 (Conflict) を応答する。

4.2.1.2.2. クラウドサービスのユーザーを更新

受け取ったユーザーデータの検証ができた場合は、受け取ったユーザーデータを用いてクラウドサービスのユーザーの情報を置き換える。

ユーザーデータとして渡されたデータの中に、クラウドサービスで使用しない属性が含まれていた場合、その情報は保管することなく破棄しなければならない。

ユーザーデータとして渡されたデータが [3.3.1 節] にあげた属性を含んでいなかった場合は、クラウドサービスが適切な属性値（空でも良い）を割り当てる。

ユーザーデータに含まれる属性「ID トークンの Issuer」(idTokenClaims.issuer)、「ID トークンの Subject」(idTokenClaims.subject) と「企業 OP でのログイン ID」(externalUserName) については、認証処理で使用する属性となるため、クラウドサービスのユーザーの属性として保管する必要がある。

ユーザー情報の更新を行なった場合、SCIM サーバーは次の 2 つの管理用属性の値を更新する必要がある。

1. バージョン

SCIM サーバーで、HTTP ETag を用いた Resource Versioning に対応する場合に、バージョンをあらわす値を生成して設定する。この値はユーザーリソースの "meta.version" 属性として用いる。この値はオプションである。

2. 更新日

このユーザーリソースを更新した日付を設定する。この値はユーザーリソースの "meta.lastModified" 属性として用いる。

更新される属性値の例を以下に示す。

| No | 属性 | 値 |
|----|-------|----------------------------------|
| 1 | バージョン | 21ad0bd836b90d08f4cf640b4c298e7c |
| 2 | 更新日 | 2015-07-01T01:02:03.004Z |

4.2.1.2.3. 更新結果の応答

ユーザーの更新が成功した場合は、HTTP ステータスコード 200 (OK) とともに、登録されたユーザーリソースを応答する。

```
HTTP/1.1 200 OK
Content-Type: application/scim+json
Location: https://scim.svc.example.net/v2/User/a2e492da-e2ed-4d90-a186-6fc01b56b8d9
ETag: W/"21ad0bd836b90d08f4cf640b4c298e7c"

{
  "schemas": [
    "urn:ietf:params:scim:schemas:core:2.0:User",
    "urn:ietf:params:scim:schemas:extension:enterprise:2.0:User",
    "urn:oidfj:params:scim:schemas:extention:enterprisejp:2.0:User"
  ],
  "id": "a2e492da-e2ed-4d90-a186-6fc01b56b8d9",
  "userName": "taro.nippon@com.example.co.jp",
  "externalId": "e1234567",
  "meta": {
    "resourceType": "User",
    "created": "2015-04-01T01:23:45.678Z",
    "lastModified": "2015-07-01T01:02:03.004Z",
    "location": "https://scim.svc.example.net/v2/User/a2e492da-e2ed-4d90-a186-6fc01b56b8d9",
    "version": "W/\\"21ad0bd836b90d08f4cf640b4c298e7c\""
  },
  ... (ユーザーデータ。以下省略) ...
}
```

クラウドサービス内の処理が理由でユーザー情報の更新中にエラーが発生した場合は、HTTP ステータスコード 500 (Internal Server Error) を応答する。このとき、利用企業の ID 管理サーバーにエラーの内容を伝達するため、`detail` にエラーの詳細を記述して応答することが望ましい。

```
HTTP/1.1 500 Internal Server Error
Content-Type: application/scim+json

{
  "schemas": ["urn:ietf:params:scim:api:messages:2.0:Error"],
  "detail": "SCIM サーバーがエラーの詳細を記入する",
  "status": "500"
}
```

SCIM を用いてユーザーを更新する場合の SCIM リクエスト・レスポンスの例を [付録 B.4] に掲載する。

4.2.1.3. ユーザーの削除 (DELETE)

4.2.1.3.1. リクエストの受け取り

ユーザーの削除は、HTTP DELETE メソッドを受け取ることで実行される。SCIM クライアントから送られてくるリクエストの例を以下に示す。

```
DELETE /Users/a2e492da-e2ed-4d90-a186-6fc01b56b8d9 HTTP/1.1
Host: scim.svc.example.net
Authorization: Basic Yzc2NTQzMjE6Z3100UpSN3pUWDRteXd0NHFiMW5ZdUhpTTA3WF10Q1U=
If-Match: W/"4124bc0a9335c27f086f24ba207a4912"
```

URI に指定されたユーザーリソースが見つからない場合は、HTTP ステータスコード 404 (Not Found) を応答する。

HTTP ETag を用いた Resource Versioning に対応している場合には、If-Match ヘッダに指定されたバージョンと、クラウドサービスで管理しているユーザー情報のバージョンが一致していることを確認する。もしバージョンが異なっている場合は、HTTP ステータスコード 409 (Conflict) を応答する。

4.2.1.3.2. クラウドサービスのユーザーを削除

クラウドサービスは、ユーザーの情報をクラウドサービスに適した方法で削除する。

SCIM の仕様では、この削除は物理削除でも論理削除でも良い。(SCIM Protocol [RFC7644] の Section 3.6) ただし、いずれの場合でも以降の SCIM 操作について DELETE が行なわれたユーザーが存在しないものとして振舞うことが要求される。つまり、HTTP DELETE メソッドを用いたユーザーの削除操作は、SCIM クライアントから見ると物理削除にであるということである。

4.2.1.3.3. 削除結果の応答

ユーザーの削除が成功した場合は、HTTP ステータスコード 204 (No Content) を応答する。
HTTP/1.1 204 No Content

クラウドサービス内の処理が理由でユーザーの削除中にエラーが発生した場合は、HTTP ステータスコード 500 (Internal Server Error) を応答する。このとき、利用企業の ID 管理サーバーにエラーの内容を伝達するため、detail にエラーの詳細を記述して応答することが望ましい。

```
HTTP/1.1 500 Internal Server Error
Content-Type: application/scim+json

{
  "schemas": ["urn:ietf:params:scim:api:messages:2.0:Error"],
  "detail": "SCIM サーバーがエラーの詳細を記入する",
  "status": "500"
}
```

SCIM を用いてユーザーを削除する場合の SCIM リクエスト・レスポンスの例を [付録 B.5] に掲載する。

4.2.2. 検索エンドポイントの実装

SCIM サーバーは、ユーザー情報のプロビジョニング処理で使用する検索エンドポイントを実装していなければならない。以降の解説では、エンドポイントの URI が以下のとおりであるものとしている。

| No | エンドポイント | URI |
|----|----------------|--|
| 1 | SCIM 検索エンドポイント | https://scim.svc.example.net/.search https://scim.svc.example.net/v2/.search |

今後の SCIM プロトコルのバージョンの変更により、エンドポイントの変更が必要となる場合がある。そのため、プロトコルのバージョンを URI に含める、SCIM Protocol Versioning に対応することが望ましい。

このエンドポイントでは、ユーザーの検索に対応する。

4.2.2.1. ユーザーの検索

4.2.2.1.1. リクエストの受け取り

ユーザーの検索は、ユーザーの更新およびユーザーの削除の際に、更新対象ユーザーの SCIM サーバー上でのユーザーリソース識別子 (id) とバージョン情報 (meta.version) を求めるために使用される。

ユーザーを特定する条件として、ID 管理サーバー上の識別子 (externalId) が用いられる。externalId が e1234567 であるユーザーリソースを問合せる場合には、filter パラメータとして externalId eq "e1234567" が指定される。

検索結果として最低限の属性値だけを応答することが望ましい。そのため応答に含める属性が attributes パラメータで指定される。attributes パラメータには、少なくとも externalId と meta が指定される。

SCIM クライアントから送られてくるリクエストの例を以下に示す。

```
POST /.search HTTP/1.1
Host: scim.svc.example.net
Accept: application/scim+json
Content-Type: application/scim+json
Authorization: Basic Yzc2NTQzMjE6Z3100UpSN3pUWDRteXd0NHFiMW5ZdUhpTTA3WFI0Q1U=
Content-Length: ...

{
  "schemas": ["urn:ietf:params:scim:api:messages:2.0:SearchRequest"],
  "attributes": ["externalId", "meta"],
  "filter": "externalId eq \"e1234567\""
}
```

このユーザーの問合せは、ユーザーエンドポイントへの HTTP GET メソッドを用いた問合せとして実装することもできる。しかし、条件式に ID 管理サーバー上の識別子 (`externalId`) を指定し、この `externalId` が個人識別可能情報 (Personally Identifiable Information: PII) に該当するため、SCIM Protocol [RFC7644] の Section 7.5 の考察により検索エンドポイントを用いる実装とする。

4.2.2.1.2. リクエストの検証

リクエストの検証として、`filter` パラメータに指定された検索条件の妥当性と、`attributes` パラメータに指定された条件の妥当性を確認する必要がある。

`filter` パラメータの検証では、条件に指定された属性が存在する属性であることの検証、SCIM Protocol [RFC7644] の仕様に沿った検索式であることの検証とともに、クラウドサービスのユーザー管理用データベースへのアクセス方法を鑑み、クエリの挿入により意図しないデータが応答されることがないことの検証も必要である。

`attributes` パラメータの検証では、条件に指定された属性が存在する属性であることの検証を行なう。

検証の結果、リクエストにエラーがあった場合は、以下のいずれかのエラーを返す。

1. リクエストメッセージに誤りがある [付録 B.3.3.2]
2. リクエストのフィルター設定に誤りがある [付録 B.3.3.3]

4.2.2.1.3. 検索結果の応答

ユーザーの問合せに成功した場合は、HTTP ステータスコード 200 (OK) とともに、検索条件に合致したユーザーリソースのリストを応答する。

```
HTTP/1.1 200 OK
Content-Type: application/scim+json

{
  "schemas": [
    "urn:ietf:params:scim:api:messages:2.0:ListResponse"
  ],
  "totalResults": 1,
  "Resources": [
    {
      "id": "a2e492da-e2ed-4d90-a186-6fc01b56b8d9",
      "externalId": "e1234567",
      "meta": {
        "resourceType": "User",
        "created": "2015-04-01T01:23:45.678Z",
        "lastModified": "2015-04-01T01:23:45.678Z",
        "location": "https://scim.svc.example.net/v2/User/a2e492da-e2ed-4d90-a186-6fc01b56b8d9",
        "version": "W/\\"4124bc0a9335c27f086f24ba207a4912\\""
      }
    }
  ]
}
```

当実装ガイドでは、ID 管理サーバー上の識別子 (`externalId`) を条件として SCIM サーバー上のユーザーリソース識別子 (`id`) を求めるための検索を行なっているため、`totalResults` が 1 である応答が正常応答であり、`totalResults` が 0 あるいは 2 以上である応答は失敗応答であるという意味を持つこととなる。

SCIM を用いてユーザーを検索する場合の SCIM リクエスト・レスポンスの例を [付録 B.3] に掲載する。

4.2.3. ユーザーデータの検証

ユーザーの作成 (POST)、ユーザーの更新 (PUT) では、SCIM クライアントからユーザーデータがリクエストボディとして送られてくる。

このデータは、次の手順で検証を行なう必要がある。

1. JSON ドキュメントとしてパースする。パースができなかった場合は、次のエラーを返す。
 - リクエストメッセージに誤りがある [付録 B.2.3.1], [付録 B.4.4.2]
2. 必須属性が含まれていることを確認する。必須属性には次の 5 つがある。
 - ID 管理サーバー上の識別子 (`externalId`)
 - サービス利用時のユーザー名 (`userName`)

- 企業 OP でのログイン ID (externalUserName)
- ID トークンの Issuer (idTokenClaims.Issuer)
- ID トークンの Subject (idTokenClaims.Subject)

必須属性が含まれていなかった場合は、次のエラーを返す。

- リクエストメッセージに誤りがある [付録 B.2.3.1], [付録 B.4.4.2]
3. クラウドサービスが使用する属性の値の検証を行なう。値の型の検証を行なうとともに、値がクラウドサービスのユーザー情報として受け入れ可能な値であるかを検証する。検証がエラーとなった場合は、次のエラーを返す。
 - ユーザー属性値に誤りがある [付録 B.2.3.2], [付録 B.4.4.3]
 4. クラウドサービス全体で一意性が必要な属性の検証を行なう。この検証は次の 3 つの属性に対して行なう。
 - サービス利用時のユーザー名 (userName) が一意である
 - ID トークンの Issuer (idTokenClaims.Issuer) と ID トークンの Subject (idTokenClaims.Subject) の組み合わせが一意である

これらの属性の値が一意とならない場合は、次のエラーを返す。

- ユーザー属性値が重複している [付録 B.2.3.3], [付録 B.4.4.4]
5. クラウドサービスの利用企業のデータの範囲（テナントのユーザー情報の範囲）で一意性が必要な属性の検証を行なう。この検証は次の 2 つの属性に対して行なう。
 - ID 管理サーバー上の識別子 (externalId)
 - 企業 OP でのログイン ID (externalUserName)

これらの属性の値が一意とならない場合は次のエラーを返す。

- ユーザー属性値が重複している [付録 B.2.3.3], [付録 B.4.4.4]

4.2.4. エンドポイントアクセス時の認証

SCIM エンドポイントアクセス時の認証方式として、少なくとも Basic 認証に対応する。Basic 認証に用いるパスワードは、クラウドサービスが十分な長さ（32 バイト以上）を持つパスワードを発行したものを使用することが推奨される。

クラウドサービスは、SCIM エンドポイントにアクセスする SCIM クライアントの IP アドレスを用いたアクセス制御、およびアクセス記録の提供を実装することが推奨される。

クラウドサービスは、SCIM クライアントの認証結果に基づいて、ユーザーリソースへのアクセスの可否、要求された操作の可否を判定しなければならない。

認証が失敗した場合には HTTP ステータスコード 401 (Unauthorized) を応答する。リソースへのアクセス、操作ができない場合は HTTP ステータスコード 403 (Forbidden) を応答する。

また、[3.6.3 節] でも述べたとおり、SCIM エンドポイントアクセス時の認証方式として、OAuth 2.0 を用いた認可、リフレッシュトークンを用いたアクセストークンの自動更新への対応を行なうことを強く推奨する。

4.3. 利用企業の管理者向け機能の実装

利用企業の管理者が、クラウドサービスと自社の認証システムの連携を設定するために必要な機能について解説する。

4.3.1. OpenID Connect OP（認証サーバー）登録機能

OpenID Connect を用いた認証連携の設定では、利用企業の認証サーバーを OpenID プロバイダーとして登録するための機能が必要となる。登録にあたり、利用企業から次の情報が提供される。

| No | 情報 | 例 |
|----|--------------------------|--|
| 1 | クラウドサービスに割り当てた client_id | pWBoRam9sG |
| 2 | Issuer の識別子 | https://op.com.example.co.jp |
| 3 | 認証エンドポイント URI | https://op.com.example.co.jp/authorize |
| 4 | ID トークン署名検証用の公開鍵 | PEM 形式 or JWK 形式の公開鍵 |
| 5 | ID トークン署名検証用の公開鍵の ID | iAw5 |

これらの情報を利用企業の管理者が Web フォーム画面へ入力し、登録管理する機能をクラウドサービスが提供することが望ましい。

このうち、ID トークン署名検証用の公開鍵に関する情報は、署名用のキーペアのローテーションのために、利用企業が適切なタイミングで更新する情報となる。そのため、公開鍵の情報の管理画面を専用を用意した方が、利用企業の管理者にとってわかりやすいものとなるだろう。

ID トークン署名検証用の公開鍵は、キーペアローテーションをスムーズに行なうため、少なくとも 2 つを登録することができる必要がある。ID トークン署名検証用の公開鍵の ID (kid) については、公開鍵毎にユニークな値が設定されていることの検証を行なうべきである。

クラウドサービスが個別の認証機能でサービスを利用する機能を有している場合、利用企業が OpenID Connect を用いた認証連携を設定することで、個別の認証機能の利用を停止する機能も必要である。

4.3.2. SCIM クライアント（ID 管理サーバー）登録機能

SCIM を用いたユーザー情報のプロビジョニングの設定では、利用企業の ID 管理サーバーを SCIM クライアントとして登録する機能が必要となる。

当実装ガイドでは、SCIM エンドポイントアクセス時の認証方式として、少なくとも Basic 認証に対応することを求めている。Basic 認証によるエンドポイントアクセスの安全性を確保するための機能として、次の 3 つを併用することを推奨する。

一つ目は、Basic 認証のためのパスワードをクラウドサービス側で生成して提供する機能である。

クラウドサービスは十分な長さ（32 バイト以上）を持つパスワードを生成し利用企業に提供する。生成したパスワードは利用企業の管理者向けの管理画面で照会する。

この機能は、利用企業の管理者が、強度が不十分なパスワードを設定して SCIM エンドポイントを利用することを予防することを狙いとするものであり、対応を強く推奨する。

二つ目は、SCIM クライアントの IP アドレスを用いたアクセス制御を行なう機能である。

Basic 認証で用いるパスワードは、自動的にかつ定期的に交換されるものではなく、人の手を介して交換・設定されるものであるため、パスワード漏えいによる第三者による SCIM エンドポイントアクセスのリスクがある。

このリスクを回避するために、クラウドサービスは SCIM クライアントの IP アドレスを用いたアクセス制御を実装することが望ましい。

SCIM クライアント登録機能として、利用企業が認識している正規の ID 管理サーバー（SCIM クライアント）の IP アドレスを事前登録することで、このアクセス制御に対応する。ID 管理サーバーを複数構成するケースもあるため、複数の IP アドレスの登録に対応する必要がある。

三つ目は、SCIM クライアントのエンドポイントアクセスの記録を提供する機能である。

SCIM エンドポイントのアクセスの記録として、クライアントの IP アドレスと行なった操作の記録を提供することで、利用企業は SCIM エンドポイントへの意図しないアクセスを検出できるようになる。

SCIM を用いたユーザー情報のプロビジョニングでは、異動情報の反映などに用いられるため、特定の処理日において膨大な回数の SCIM エンドポイントの呼び出しが行なわれることとなる。

そのため、利用企業の管理者向けに提供されるアクセスの記録では、全アクセスの記録だけではなく、単位期間毎のアクセス元 IP 毎のアクセス数の情報などのサマリ情報の提供を行なうと良い。

4.3.3. 管理者向け機能利用時の認証方式

利用企業の管理者向け機能は、認証連携にかかわる重要な設定を扱うものとなる。そのため、管理者向け機能利用時の認証には、ID とパスワードによる認証に加え、ワンタイムパスワードを利用するなど、多要素によるより安全な方式を提供すべきである。

第5章 クラウドサービス利用企業向け実装ガイド

5.1. OpenID Connect OP（認証サーバー）の実装

この節では、クラウドサービスへユーザーの認証結果を連携する OpenID Connect OpenID プロバイダーとしての実装について解説する。

5.1.1. 認証フローの概要

認証フローは OpenID Connect の Implicit フローを用いる。実装する認証フローの全体を以下に示す。

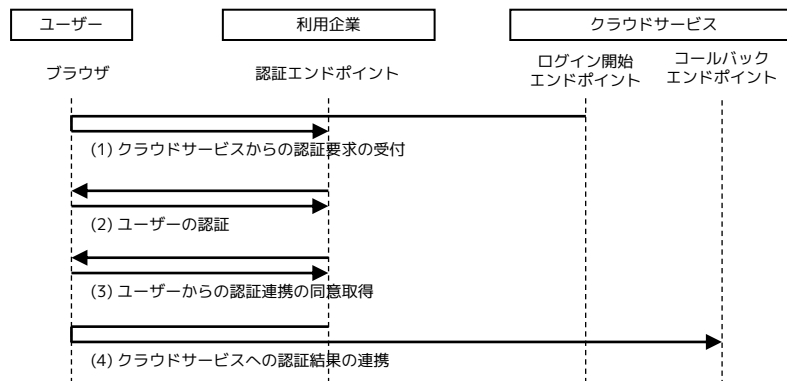


図 5-1: OpenID Connect Implicit フローの OP 処理シーケンス

5.1.2. クラウドサービスから認証要求を受け取る

5.1.2.1. 通常の認証要求

クラウドサービスからの認証要求を受け取るため、認証エンドポイントを実装する。認証エンドポイントでは次のパラメータをクエリストリングの形式で受け取る。

| No | パラメータ | 例 | 説明 |
|----|---------------|--|--|
| 1 | response_type | id_token | Implicit フローで ID トークンを受け取るため "id_token" を指定する |
| 2 | client_id | pWBoRam9sG | クラウドサービスに割り当てた ID |
| 3 | redirect_uri | https%3A%2F%2Fwww.svc.example.net%2Fcb | 認証後にリダイレクトする先のクラウドサービスの URI。フラグメントに ID トークンがついた状態でリダイレクトされる。 |
| 4 | scope | openid | OpenID Connect を使用するため "openid" を指定する |

| | | | |
|---|-------|--------------|-------------------------------|
| 5 | state | k4y97klszxi | 認証要求と認証後のリダイレクトアクセスを関連付ける文字列 |
| 6 | nonce | q8k-upBX4Z_A | クライアントセッションと ID トークンを関連付ける文字列 |

認証サーバーは受け取ったパラメータについて、次の検証を行なう。

1. response_type パラメータ

Implicit フローを使うため、"id_token" もしくは "id_token token" のいずれかであることを検証する。

当実装ガイドでは ID トークンだけを必要としているため、"id_token" が指定されていれば十分である。しかし、クラウドサービスが response_type に "token" を含めてしまう可能性もあるため "id_token token" も許容する。

2. client_id パラメータ

認証サーバーに登録されたクラウドサービスの client_id であることを検証する。

3. redirect_uri パラメータ

渡された URI が認証サーバーに登録されたクラウドサービスのリダイレクト先 URI の一覧に含まれていることを検証する。

このときに行なう URI の比較は、クエリストリングが付いている場合にはそれも含めた完全一致で行なわなければならない。

比較を完全一致で行なわなかった場合、認証サーバー自体がオープンリダイレクタとなる可能性があり、ID トークンの流出、盗用、それに伴うクラウドサービスの不正利用の発生につながることに注意が必要である。

4. scope パラメータ

OpenID Connect を使用するため "openid" が指定されていることを検証する。

当実装ガイドでは、認証結果のみの連携を扱うため、ユーザーの属性は連携しないよう "openid" のみが指定されていることを検証することを推奨する。

クラウドサービスにユーザーの属性を連携する場合は、"openid" とともに適切なスコープの値が指定されていることを検証しなければならない。

5. state パラメータ

state パラメータの指定は推奨されるが、必須パラメータではない。

6. nonce パラメータ

nonce パラメータが指定されていることを検証する。

認証サーバーが上記以外のパラメータを受け取った場合、そのパラメータを適切に処理するか、あるいは存在しないものとして処理をしなければならない。

パラメータの検証に失敗した場合は、[5.1.5.2 節]の方法でエラーを応答する。

5.1.2.2. 再認証のための認証要求

クラウドサービスが再認証のための認証要求を行なった場合は、通常の認証要求のパラメータに加え、次のパラメータを受け取る。

| No | パラメータ | 例 | 説明 |
|----|------------|----------|--|
| 1 | prompt | login | 認証済みであっても認証画面を必ず表示する。"login" を指定する。 |
| 2 | max_age | 30 | ID トークンに auth_time クレームを含める。30 秒を指定する。 |
| 3 | login_hint | e1234567 | 企業 OP でのログイン ID (externalUserName) としてプロビジョニングされた値 |

認証サーバーは受け取ったパラメータについて、次の検証を行なう。

1. prompt パラメータ

prompt パラメータとして、"none", "login", "consent", "select_account" のいずれかの値、あるいはスペースで区切られた複数の値であることを検証する。

2. max_age パラメータ

秒数を表す整数の値であることを検証する。

3. login_hint パラメータ

ログイン ID として適切な文字列であることを検証する。

ログイン ID と一致することの検証は不要である。ログイン ID と一致しないことを明示的なエラーとして応答することで、ログイン ID の存在確認に用いられる可能性がある点に注意が必要である。

パラメータの検証に失敗した場合は、[5.1.5.2 節]の方法でエラーを応答する。

5.1.3. ユーザーを認証する

利用企業のポリシー、認証サーバーの実装に従って、ユーザーの認証を行なう。

通常の認証要求の場合は、ユーザーが認証サーバーで認証済みの場合は、ユーザーに認証画面を表示する必要はない。

認証要求に `prompt=login` のパラメータが指定されている場合は、再認証要求のため、ユーザーが認証済みであるか否かにかかわらず、ユーザーに認証画面を表示し、明示的な認証処理を行なう必要がある。

認証要求の中に `login_hint` パラメータが指定されている場合は、認証画面を表示する際に、ログイン ID の初期値として `login_hint` パラメータに指定された値を表示すべきである。

認証要求に `prompt` パラメータとして `"none"`, `"consent"`, `"select_account"` が指定された場合の動作は OpenID Connect Core 1.0 [OpenID.Core] の Section 3.1.2.1 に従うべきである。

5.1.4. ユーザーに認証連携の同意を得る

ユーザーの認証ができた場合、認証サーバーはクラウドサービスへ認証結果の情報提供について、ユーザーに同意を得る。

OpenID Connect での一般的な同意/許可の取得方法は、ユーザーに同意画面を提示しインタラクティブに行なう方法であるが、管理機能を用いた事前同意など、その他の方法を用いた同意/許可の取得方法を用いることも選択できる。(OpenID Connect Core 1.0 [OpenID.Core] の Section 3.1.2.4)

企業がクラウドサービスを利用する場合、ユーザー（従業員）の認証結果の情報提供は企業が判断すべきものであり、個々のユーザーに同意を求めるものではない。

一方、認証結果以外のユーザーの属性情報については、以下の 2 つの考え方がありえる。

1. 企業が従業員に付与した属性情報の提供については、企業としての判断で行なうべきものであり、ユーザー（従業員）への明示的な同意取得は不要である。
2. 企業が従業員に付与した属性情報についても、個人のプライバシーにかかわるデータであり、クラウドサービスに提供する前にユーザー（従業員）への明示的な同意の取得が必要である。

この考え方は、利用するクラウドサービスの特性によっても異なる。

以上を踏まえると、ユーザーの同意取得の推奨される実装方法としては、次のようなものが考えられる。

1. クラウドサービスへ認証結果だけを連携する場合、すなわち認証要求時にスコープとして "openid" のみが指定されている場合は、同意画面の表示は行なわない。
2. 認証要求時にスコープとして "openid" 以外のものが指定されている場合に同意画面の表示を行なうか否かは、企業のポリシーおよび利用するクラウドサービスに応じて選択する。

5.1.5. クラウドサービスへ認証結果を返す

5.1.5.1. 認証成功時の応答

ユーザー認証に成功した場合は、認証要求時に `redirect_uri` パラメータで指定された URI に、フラグメントとして以下のパラメータからなるクエリストリングを連結した URI へリダイレクトする。

| No | パラメータ | 例 | 説明 |
|----|----------|---|------------------------------------|
| 1 | id_token | eyJhbGciOiAi... .eyJpc3MiOiAi... .rZR3EjzaHA... | ID トークン [5.1.6 節] |
| 2 | state | k4y97klszxi | 認証要求時に state パラメータが渡されたときはその値を設定する |

応答の例を以下に示す。

```
HTTP/1.1 302 Found
Location: https://www.svc.example.net/cb#
  id_token=eyJhbGciOiAi... .eyJpc3MiOiAi... .rZR3EjzaHA...
  &state=k4y97klszxi
```

5.1.5.2. 認証失敗時の応答

ユーザー認証に失敗した場合は、認証要求時に `redirect_uri` パラメータで指定された URI に、フラグメントとして以下のパラメータからなるクエリストリングを連結した URI へリダイレクトする。

| No | パラメータ | 例 | 説明 |
|----|-------------------|-----------------------------|---|
| 1 | error | access_denied | OAuth 2.0 [RFC6749] の Section 5.2, OpenID Connect Core 1.0 [OpenID.Core] の Section 3.1.2.6 にあげられるエラーコード |
| 2 | error_description | User authentication failed. | エラー理由をあらわす ASCII テキスト |

| | | | |
|---|-------|-------------|------------------------------------|
| 3 | state | k4y97klszxi | 認証要求時に state パラメータが渡されたときはその値を設定する |
|---|-------|-------------|------------------------------------|

応答の例を以下に示す。

```
HTTP/1.1 302 Found
Location: https://www.svc.example.net/cb#
  error=access_denied
  &error_description=User%20authentication%20failed.
  &state=k4y97klszxi
```

OpenID Connect を用いてユーザーの認証連携を行なう場合の OpenID Connect のリクエスト・レスポンスの例を [付録 A.2], [付録 A.3] に掲載する。

5.1.6. ID トークン

認証サーバーでの認証結果は、ID トークンの形でクラウドサービスへ返す。ID トークンは、JWS (JSON Web Signature) で署名された JWT (JSON Web Token) [RFC7519] の形式で作成する。

5.1.6.1. JOSE ヘッダ

ID トークンの 1 番目のパートは JOSE ヘッダである。JOSE ヘッダには少なくとも次の 2 つの属性を設定する。

| No | 属性 | 例 | 説明 |
|----|-----|-------|--------------------------------------|
| 1 | alg | RS256 | 署名アルゴリズム。当実装ガイドでは RS256 への対応を指定している。 |
| 2 | kid | iAw5 | 署名検証用の公開鍵の ID |

JOSE ヘッダの例を以下に示す。

```
{
  "alg": "RS256",
  "kid": "iAw5"
}
```

これを base64url エンコーディングしたものが、ID トークンの 1 番目のパートとなる。

```
eyJhbGciOiJSUzI1NiIsImtpZCI6IiAw5"}
```


5.1.6.2. ID トークンに格納するクレーム

ID トークンの 2 番目のパートは、ID トークンのクレームとなる。少なくとも [3.4.1 節] に示したクレームを含める。

ID トークンのクレームの例を以下に示す。通常の認証要求の場合は `auth_time` を含まない ID トークンを返す。(含まれていても良い)

```
{
  "iss": "https://op.com.example.co.jp",
  "sub": "e1234567",
  "aud": "pWBoRam9sG",
  "nonce": "q8k-upBX4Z_A",
  "exp": 1435709162,
  "iat": 1435708862
}
```

再認証要求の場合は、認証要求に `max_age` パラメータが指定されているため、`auth_time` を含めた ID トークンを返さなければならない。

これを `base64url` エンコーディングしたものが、ID トークンの 2 番目のパートとなる。

```
eyJJpc3MiOiJodHRwczovL29wLmNvbS5leGFtcGxlLmNvLmpwIiwic3ViIjoizTEyMzQ1NjciLCJhdWQiOiJwV0JvUmFtOXBhcnR1eWUyIiwiaWF0IjoxNDM1NzA5MTYyLCJleHAiOjE0MzU3MDkxNjIsIm1hdCI6MTQzNTcwODg2Mn0
```

5.1.6.3. ID トークンの署名

ID トークンの `base64url` エンコーディングされた 1 番目、2 番目のパートを `!` で連結した文字列を、ハッシュ関数に `SHA-256` を使うように構成された `RSASSA-PKCS1-v1_5` 署名器にかけることで署名が行なえる。

このとき署名には秘密鍵を使用する。

OpenID Connect の ID トークンの例を [付録 A.1] に掲載する。

5.1.7. ユーザーセッションを強制ログアウトする

ユーザーのセッションを強制ログアウトする場合は、OpenID Connect Back-Channel Logout 1.0 [OpenID.Backchannel] を使い、クラウドサービスが提供するログアウトエンドポイントへログアウトトークンを送信する。

5.1.7.1. ログアウトトークン

ログアウトトークンには、少なくとも [3.5.1 節] に示したクレームを含めなければならない。

ログアウトトークンには、[3.5.1 節] で述べたとおり nonce クレームを含めてはならない。

ログアウトトークンのクレームの例を以下に示す。

```
{
  "iss": "https://op.com.example.co.jp",
  "sub": "e1234567",
  "aud": "pWBoRam9sG",
  "exp": 1435733336,
  "jti": "5ByK",
  "logout_only": true
}
```

jti クレームは、このログアウトトークンを表すユニークな文字列を指定する。

ログアウトトークンの再利用を予防するため、exp の値はログアウトトークンの発行時間から 2 分以内とすることが推奨されている。

このクレームに、ID トークンと同じ方法 [5.1.6 節] を用いて JOSE ヘッダと署名をつけることで、ログアウトトークンとなる。

OpenID Connect Back-Channel Logout のログアウトトークンの例を [付録 A.4] に掲載する。

5.1.7.2. クラウドサービスへのログアウトの要求

クラウドサービスのログアウトエンドポイントへ HTTP POST を用いてログアウトトークンを送ることで、ログアウトを要求することができる。

```
POST /bc_logout HTTP/1.1
Host: www.svc.example.net
Content-Type: application/x-www-form-urlencoded

logout_token=eyJhbGciOiAi... .eyJpc3MiOiAi... .zPt-EH3b9m ...
```

ログアウト要求が正しく処理された場合は、HTTP ステータスコード 200 (OK) が応答される。

ログアウトの処理中にエラーが発生した場合は、その内容により 400 (Bad Request)、501 (Not Implemented)、504 (Gateway Timeout) のいずれかの HTTP ステータスコードが応答される。

5.1.7.3. ログアウト要求を送るクラウドサービスの選択方法

当実装ガイドでは、強制ログアウトが必要なユースケースとして、「ユーザーが使用している端末を紛失したなどで、緊急でユーザーのセッションをすべて停止する必要がある場合」を想定している。

そのため、ログアウト要求は、ユーザーが利用中の全クラウドサービスを対象に送信する必要がある。

ログアウト要求を送信するクラウドサービスは、利用企業の認証サーバーでユーザー毎に認証連携済みクラウドサービスが管理できる場合は、そのクラウドサービスに限定するほうが望ましい。

利用企業の認証サーバーでユーザー毎に認証連携済みクラウドサービスを管理することが困難である場合は、利用企業の認証サーバーに接続されたクラウドサービスすべてにログアウト要求を送信すべきである。

強制ログアウトが必要なユースケースの特性から、強制ログアウト要求は、複数のクラウドサービスへ並行して送信することが推奨される。

ログアウト処理中にエラーが発生した場合、リトライで完全に解決ができない場合がある。よってログアウト処理でエラーが発生した場合のセキュリティリスクの検討と適切な代替の対応手順の検討をすべきである。

5.1.8. 公開鍵の公開（JWK Set エンドポイントの実装）

署名アルゴリズムに RS256 を使用する場合、署名に使用するキーペアを準備しなければならない。キーペアのうち、公開鍵はクラウドサービス事業者へ提供する必要がある。その方法は 2 つある。

1. クラウドサービスが提供する利用企業の管理者向け機能で、認証サーバーの登録画面から署名検証用の公開鍵を登録する。
2. 利用企業が署名用公開鍵を提供するエンドポイントを実装し、その URI をクラウドサービスへ `jwtks_uri` として設定し、そのエンドポイントから公開鍵をクラウドサービスへ提供する。

署名用のキーペアは適切なタイミングでローテーションを行なう必要がある。キーペアのローテーションを行なった場合、方法 1 を用いた公開鍵の提供方法では、利用するクラウドサービスが増えれば増えるほど、鍵更新の手間が増えてしまう課題がある。

そのため、方法 2 の `jwtks_uri` を用いた公開鍵の提供方式を用いることを推奨する。`jwtks_uri` 自体は、認証サーバーと異なるサーバーで提供してよい。ただし、信頼できるサーバーから公開鍵が提供されいることを証明できるよう、`jwtks_uri` は `https` スキームで始まる URI であるべきである。

`jwtks_uri` で提供する公開鍵情報は JWK Set 形式 (JSON Web Key [RFC7517] の Section 5) を用いる。JWK Set 形式の公開鍵情報の例を [付録 A.6] に掲載する。

5.2. SCIM クライアントの実装

この節では、利用企業の ID 管理サーバーが、SCIM を使ってクラウドサービスへユーザー情報のプロビジョニングを行なう実装について解説する。

5.2.1. ユーザー情報の操作

ユーザー情報をプロビジョニングする場合は、1 ユーザーごとに SCIM の操作を行なう。

5.2.1.1. ユーザーの作成

ユーザーを作成する場合は、SCIM サーバーの User エンドポイントに対して、JSON で表現されたユーザー情報を HTTP POST する。リクエストの例を以下に示す。

```
POST /Users HTTP/1.1
Host: scim.svc.example.net
Accept: application/scim+json
Content-Type: application/scim+json
Authorization: Basic Yzc2NTQzMjE6Z3l0OUUpSN3pUWDRteXd0NHFiMW5ZdUhpTTA3WF10Q1U=
Content-Length: ...

{
  "schemas": [
    "urn:ietf:params:scim:schemas:core:2.0:User",
    "urn:ietf:params:scim:schemas:extension:enterprise:2.0:User",
    "urn:oidfj:params:scim:schemas:extention:enterprisejp:2.0:User"
  ],
  "userName": "taro.nippon@com.example.co.jp",
  "externalId": "e1234567",
  ... (ユーザーデータ。以下省略) ...
}
```

ユーザーが正しく登録された場合は、HTTP ステータスコード 201 (Created) とともに、登録されたユーザーリソースが応答される。

このユーザーリソースは、リクエストしたユーザー情報にリソース識別子 (id) とメタ情報 (meta) が付加されたものとなっている。

```
HTTP/1.1 201 Created
Content-Type: application/scim+json
Location: https://scim.svc.example.net/v2/User/a2e492da-e2ed-4d90-a186-6fc01b56b8d9
ETag: W/"4124bc0a9335c27f086f24ba207a4912"

{
  "schemas": [
    "urn:ietf:params:scim:schemas:core:2.0:User",
    "urn:ietf:params:scim:schemas:extension:enterprise:2.0:User",
    "urn:oidfj:params:scim:schemas:extension:enterprisejp:2.0:User"
  ],
  "id": "a2e492da-e2ed-4d90-a186-6fc01b56b8d9",
  "userName": "taro.nippon@com.example.co.jp",
  "externalId": "e1234567",
  "meta": {
    "resourceType": "User",
    "created": "2015-04-01T01:23:45.678Z",
    "lastModified": "2015-04-01T01:23:45.678Z",
    "location": "https://scim.svc.example.net/v2/User/a2e492da-e2ed-4d90-a186-6fc01b56b8d9",
    "version": "W/\\"4124bc0a9335c27f086f24ba207a4912\""
  },
  ... (ユーザーデータ。以下省略) ...
}
```

ETag ヘッダ、および「バージョン」(meta.version) 属性は、SCIM サーバーが Resource Versioning に対応している場合にのみ応答される。

SCIM サーバーからのエラーの応答については、[5.2.1.4 節] で解説する。

SCIM を用いてユーザーを作成する場合の SCIM リクエスト・レスポンスの例を [付録 B.2] に掲載する。

5.2.1.2. ユーザーの更新

ユーザーの更新を行なう場合は、対象ユーザーのユーザーリソースに対して、JSON で表現されたユーザー情報を HTTP PUT する。

更新対象のユーザーリソースを指定するために、SCIM サーバー上でのユーザーリソースの識別子 (id) が必要となる。そのため、ユーザーの更新は以下の 2 つのステップで実行する必要がある。

1. ユーザー検索によるユーザーリソース識別子 (id) の取得
2. HTTP PUT によるユーザー情報の更新

5.2.1.2.1. ユーザーリソース識別子 (id) の取得

ユーザーリソース識別子を取得するために、ID 管理サーバー上の識別子 (externalId) を条件としてユーザーリソースを検索する。

```
POST /.search HTTP/1.1
Host: scim.svc.example.net
Accept: application/scim+json
Content-Type: application/scim+json
Authorization: Basic Yzc2NTQzMjE6Z3l0OUUpSN3pUWDRteXd0NHFiMW5ZdUhpTTA3WFI0Q1U=
Content-Length: ...

{
  "schemas": ["urn:ietf:params:scim:api:messages:2.0:SearchRequest"],
  "attributes": ["externalId", "meta"],
  "filter": "externalId eq \"e1234567\""
}
```

該当ユーザーが見つければ、次のとおり 1 件のユーザーリソースが応答される。一意性を持つ属性 externalId を用いた検索のため、検索が成功した場合は必ず 1 件 (totalResults が 1) の応答となる。

```
HTTP/1.1 200 OK
Content-Type: application/scim+json

{
  "schemas": [
    "urn:ietf:params:scim:api:messages:2.0:ListResponse"
  ],
  "totalResults": 1,
  "Resources": [
    {
      "id": "a2e492da-e2ed-4d90-a186-6fc01b56b8d9",
      "externalId": "e1234567",
      "meta": {
        "resourceType": "User",
        "created": "2015-04-01T01:23:45.678Z",
        "lastModified": "2015-04-01T01:23:45.678Z",
        "location": "https://scim.svc.example.net/v2/User/a2e492da-e2ed-4d90-a186-6fc01b56b8d9",
        "version": "W/\"4124bc0a9335c27f086f24ba207a4912\""
      }
    }
  ]
}
```

この結果の Resources 属性の 1 番目の要素の id 属性の値が、更新対象のユーザーのリソース識別子 (id) となる。

また、SCIM サーバーが Resource Versioning に対応している場合は、meta.version 属性の値が、現在のリソースのバージョンとなる。

SCIM を用いてユーザーを検索する場合の SCIM リクエスト・レスポンスの例を [付録 B.3] に掲載する。

5.2.1.2.2. ユーザー情報の更新

更新対象ユーザーのユーザーリソースの URI は、SCIM サーバーの User エンドポイント URI と "/" とリソース識別子 (id) を連結したものとなる。

このエンドポイントに対して、JSON で表現されたユーザー情報を HTTP PUT することで、ユーザー情報を更新することができる。HTTP PUT はユーザー情報を置き換える操作のため、SCIM クライアントは変更の有無にかかわらず、ユーザーデータの全属性を送る必要がある点に注意が必要である。

SCIM サーバーが Resource Versioning に対応している場合は、どのバージョンのユーザーリソースへの変更であるかを示すため、リクエストヘッダに If-Match ヘッダを追加する。

```
PUT /Users/a2e492da-e2ed-4d90-a186-6fc01b56b8d9 HTTP/1.1
Host: scim.svc.example.net
Accept: application/scim+json
Content-Type: application/scim+json
Authorization: Basic Yzc2NTQzMjE6Z3100UpSN3pUWDRteXd0NHFiMW5ZdUhpTTA3WF10Q1U=
Content-Length: ...
If-Match: W/"4124bc0a9335c27f086f24ba207a4912"

{
  "schemas": [
    "urn:ietf:params:scim:schemas:core:2.0:User",
    "urn:ietf:params:scim:schemas:extension:enterprise:2.0:User",
    "urn:oidfj:params:scim:schemas:extention:enterprisejp:2.0:User"
  ],
  "id": "a2e492da-e2ed-4d90-a186-6fc01b56b8d9",
  "userName": "taro.nippon@com.example.co.jp",
  "externalId": "e1234567",
  ... (更新内容を含むユーザーデータ。以下省略) ...
}
```

ユーザー情報が正しく更新された場合は、HTTP ステータスコード 200 (OK) とともに、登録されたユーザーリソースが応答される。

このユーザーリソースは、リクエストしたユーザー情報にリソース識別子 (id) とメタ情報 (meta) が付加されたものとなっている。情報の更新によりメタ情報の中の "lastModified" 属性と "version" 属性の値も更新される。

```
HTTP/1.1 200 OK
Content-Type: application/scim+json
Location: https://scim.svc.example.net/v2/User/a2e492da-e2ed-4d90-a186-6fc01b56b8d9
ETag: W/"21ad0bd836b90d08f4cf640b4c298e7c"

{
  "schemas": [
    "urn:ietf:params:scim:schemas:core:2.0:User",
    "urn:ietf:params:scim:schemas:extension:enterprise:2.0:User",
    "urn:oidfj:params:scim:schemas:extension:enterprisejp:2.0:User"
  ],
  "id": "a2e492da-e2ed-4d90-a186-6fc01b56b8d9",
  "userName": "taro.nippon@com.example.co.jp",
  "externalId": "e1234567",
  "meta": {
    "resourceType": "User",
    "created": "2015-04-01T01:23:45.678Z",
    "lastModified": "2015-07-01T01:02:03.004Z",
    "location": "https://scim.svc.example.net/v2/User/a2e492da-e2ed-4d90-a186-6fc01b56b8d9",
    "version": "W/\\"21ad0bd836b90d08f4cf640b4c298e7c\""
  },
  ... (ユーザーデータ。以下省略) ...
}
```

SCIM サーバーからのエラーの応答については、[5.2.1.4 節] で解説する。

SCIM を用いてユーザーを更新する場合の SCIM リクエスト・レスポンスの例を [付録 B.4] に掲載する。

5.2.1.3. ユーザーの削除

ユーザーの削除を行なう場合は、対象ユーザーのユーザーリソースに対して、HTTP DELETE をリクエストする。

削除対象のユーザーリソースを指定するために、SCIM サーバー上でのユーザーリソースの識別子 (id) が必要となる。そのため、ユーザーの削除は以下の 2 つのステップで実行する必要がある。

1. ユーザー検索によるユーザーリソース識別子 (id) の取得
2. HTTP DELETE によるユーザー情報の削除

ユーザーリソース識別子 (id) の取得は、[5.2.1.2.1 節] に示した手順で行なう。

削除対象ユーザーのユーザーリソースの URI は、SCIM サーバーの User エンドポイント URI と "/" とリソース識別子 (id) を連結したものとなる。

このエンドポイントに対して、HTTP DELETE をリクエストすることで、ユーザーを削除することができる。

SCIM サーバーが Resource Versioning に対応している場合は、どのバージョンのユーザーリソースの削除であるかを示すため、リクエストヘッダに If-Match ヘッダを追加する。

```
DELETE /Users/a2e492da-e2ed-4d90-a186-6fc01b56b8d9 HTTP/1.1
Host: scim.svc.example.net
Authorization: Basic Yzc2NTQzMjE6Z3100UpSN3pUWDRteXd0NHFiMW5ZdUhpTTA3WF10Q1U=
If-Match: W/"4124bc0a9335c27f086f24ba207a4912"
```

ユーザーが正しく削除された場合は、HTTP ステータスコード 204 (No Content) が応答される。

```
HTTP/1.1 204 No Content
```

SCIM サーバーからのエラーの応答については、[5.2.1.4 節] で解説する。

SCIM を用いてユーザーを削除する場合の SCIM リクエスト・レスポンスの例を [付録 B.5] に掲載する。

5.2.1.4. エラー応答

SCIM サーバーへのリクエストにエラーが生じた場合は、それに対応するエラーが返される。

エラー応答では、エラー内容に合わせた HTTP ステータスコードと、リクエストボディとして JSON 形式のエラーメッセージが返される。

主要なエラーとして次のようなものがある。

| No | エラー内容 | コード | scimType |
|----|----------------------|-----|---------------|
| 1 | 認証に失敗した | 401 | — |
| 2 | リソースへのアクセス・操作権限がない | 403 | — |
| 3 | ユーザーリソースが見つからない | 404 | — |
| 4 | リクエストメッセージに誤りがある | 400 | invalidSyntax |
| 5 | ユーザー属性値に誤りがある | 400 | invalidValue |
| 6 | ユーザー属性値が重複している | 400 | uniqueness |
| 7 | リクエストのフィルター設定に誤りがある | 400 | invalidFilter |
| 8 | ユーザーの更新 (削除) 要求が競合した | 409 | — |
| 9 | リクエスト処理中にエラーが発生した | 500 | — |

HTTP ステータスコード 400 の応答の場合、エラーメッセージの "scimType" 属性にエラーの種類が応答される。

エラーメッセージには "detail" 属性があり、SCIM サーバーがエラーの内容、詳細を応答して与えることがある。

5.2.2. エンドポイントアクセス時の認証

SCIM エンドポイントアクセス時の認証方式として、少なくとも Basic 認証に対応する。Basic 認証に用いるパスワードは、クラウドサービスが発行した十分な長さ（32 バイト以上）を持つパスワードを使用することが推奨される。このパスワードは、クラウドサービスが提供する利用企業の管理者向けの管理画面で照会する。

参考文献

- [OpenID.Core] Sakimura, N., Bradley, J., Jones, M., de Medeiros, B., and C. Mortimore, "OpenID Connect Core 1.0", November 2014.
- [OpenID.Backchannel] Jones, M., "OpenID Connect Back-Channel Logout 1.0", September 2015.
- [OpenID.Session] Sakimura, N., Bradley, J., Jones, M., de Medeiros, B., Mortimore, C., and E. Jay, "OpenID Connect Session Management 1.0", August 2015.
- [OpenID.Logout] Jones, M., "OpenID Connect HTTP-Based Logout 1.0", September 2015.
- [RFC7643] Hunt, P., Ed., Grizzle, K., Wahlstroem, E., and C. Mortimore, "System for Cross-domain Identity Management: Core Schema", RFC 7643, September 2015.
- [RFC7644] Hunt, P., Ed., Grizzle, K., Ansari, M., Wahlstroem, E., and C. Mortimore, "System for Cross-domain Identity Management: Protocol", RFC 7644, September 2015.
- [RFC6749] Hardt, D., Ed., "The OAuth 2.0 Authorization Framework", RFC 6749, October 2012.
- [RFC6750] Jones, M. and D. Hardt, "The OAuth 2.0 Authorization Framework: Bearer Token Usage", RFC 6750, October 2012.
- [RFC7515] Jones, M., Bradley, J., and N. Sakimura, "JSON Web Signature (JWS)", RFC 7515, May 2015.
- [RFC7516] Jones, M. and J. Hildebrand, "JSON Web Encryption (JWE)", RFC 7516, May 2015,
- [RFC7517] Jones, M., "JSON Web Key (JWK)", RFC 7517, May 2015.
- [RFC7519] Jones, M., Bradley, J., and N. Sakimura, "JSON Web Token (JWT)", RFC 7519, May 2015.
- [RFC5646] Phillips, A., Ed., and M. Davis, Ed., "Tags for Identifying Languages", BCP 47, RFC 5646, September 2009.
- [XML-Schema] Peterson, D., Gao, S., Malhotra, A., Sperberg-McQueen, C., and H. Thompson, "XML Schema Definition Language (XSD) 1.1 Part 2: Datatypes", April 2012.

付録A. OpenID Connect リクエスト・レスポンス例

A.1. OpenID Connect ID トークン

A.1.1. 通常の認証要求で応答される ID トークン

A.1.1.1. JOSE ヘッダ

```
{  
  "alg": "RS256",  
  "kid": "iAw5"  
}
```

A.1.1.2. クレーム

```
{  
  "iss": "https://op.com.example.co.jp",  
  "sub": "e1234567",  
  "aud": "pwBoRam9sG",  
  "nonce": "q8k-upBX4Z_A",  
  "exp": 1435709162,  
  "iat": 1435708862  
}
```

A.1.1.3. ID トークン

```
eyJhbGciOiJSUzI1NiIsImtpZCI6Im1BdzUifQ.eyJpc3MiOiJodHRwczovL29wLmNvbS5leGFtcGxlLmNvLmpwIiwic3ViIjoizTEyMzQ1NjciLCJhdWQiOiJwV0JvUmFtOjEwIiwibm9uY2UiOiJxOGstdXBWDRaX0EiLCJleHAiOjE0MzU3MDkxNjIsIm1hdCI6MTQzNTcwODg2Mn0.rZR3EjzaHA_BCXiIkD0KctiFYnRd7hZriG36-Yx7iEckVPG2srQ0nNIP9u5KHIG6qzhHu2PBR8tjVYU2aldL8pf-VNAhD3AMYYju_TQ2qXxp5f5o08VuqX7rc-vV4hjXcwPR8CUsXveJwpLpI1TPcZyMIDkBVZNCWPRDgtAJwDYtLmFX-ONf1Y9sMvwdVBBaNFxrIcLXPwb5bhtDGM717KW1rwnrVo11av6KZUuGaxuwUqQt085GftsfgMbh3cj9W5xEH9QOKULy4C2r6qKIydCzw99WkUnoU1fDz50mjSdLGJnfCo1xYZjvJnyoAu3BUu0knE7sNpzDwn4DxJLVhHg
```

A.1.2. 再認証要求で応答される ID トークン

A.1.2.1. JOSE ヘッダ

```
{  
  "alg": "RS256",  
  "kid": "iAw5"  
}
```

A.1.2.2. クレーム

```
{
  "iss": "https://op.com.example.co.jp",
  "sub": "e1234567",
  "aud": "pWBoRam9sG",
  "nonce": "q8k-upBX4Z_A",
  "exp": 1435710062,
  "iat": 1435709762,
  "auth_time": 1435709762
}
```

A.1.2.3. ID トークン

```
eyJhbGciOiJSUzI1NiIsImtpZCI6Im1BdzUifQ.eyJpc3MiOiJodHRwczovL29wLmNvbS5leGFtcGxlLmNvLmpwIiwic3ViIjoizTEyMzQ1NjciLCJhdWQiOiJwV0JvUmFtOXBHImIiwibm9uY2UiOiJxOGstdXBWDRaX0EiLCJleHAiOjE0MzU3MTAwNjIsIm1hdCI6MTQzNTcwOTc2MiwiaXYXV0aF90aW11IjoxNDM1NzA5NzYyfQ.paxubCBAPwITlNgg-Mi_RkX5EfaRVU8YGTZ2e9UwUX1EIwBDU3i_uCqUj-yUMY6Li1rjbpHurYEw7N3ZQPdBlVy6GiMQtUFg6Ju-0MY4rw0uiZ7HFoGenAMzoR8fNd4iIuyM4o0EF6WuV5JLfZmJ5fvgjTbAD5H-2SGeM38P8UibRyv2lB-YldI8a7nEUGxz5m5iw-WpBgk4SboMwXsyg-jr-_fbRMn9_RR76-691P45-1p8BU8wBMEwghVARTbRg53W6d1ILAL_FDWWIBxUboI9_uTKu7HCYuZecU7Uub6MXG5EMWJKUPjVuuHuvkzBs17MdKCRZuwI1fEAU351YQ
```

A.2. OpenID Connect による認証

A.2.1. 認証リクエスト

```
GET /authorize?
  response_type=id_token
  &client_id=pWBoRam9sG
  &redirect_uri=https%3A%2Fwww.svc.example.net%2Fcb
  &scope=openid
  &state=k4y97klszxi
  &nonce=q8k-upBX4Z_A HTTP/1.1
Host: op.com.example.co.jp
```

A.2.2. 認証成功レスポンス

```
HTTP/1.1 302 Found
Location: https://www.svc.example.net/cb#
  id_token=eyJhbGciOiJSUzI1NiIsImtpZCI6Im1BdzUifQ.eyJpc3MiOiJodHRwczovL29wLmNvbS5leGFtcGxlLmNvLmpwIiwic3ViIjoizTEyMzQ1NjciLCJhdWQiOiJwV0JvUmFtOXBHImIiwibm9uY2UiOiJxOGstdXBWDRaX0EiLCJleHAiOjE0MzU3MTAwNjIsIm1hdCI6MTQzNTcwOTc2MiwiaXYXV0aF90aW11IjoxNDM1NzA5NzYyfQ.paxubCBAPwITlNgg-Mi_RkX5EfaRVU8YGTZ2e9UwUX1EIwBDU3i_uCqUj-yUMY6Li1rjbpHurYEw7N3ZQPdBlVy6GiMQtUFg6Ju-0MY4rw0uiZ7HFoGenAMzoR8fNd4iIuyM4o0EF6WuV5JLfZmJ5fvgjTbAD5H-2SGeM38P8UibRyv2lB-YldI8a7nEUGxz5m5iw-WpBgk4SboMwXsyg-jr-_fbRMn9_RR76-691P45-1p8BU8wBMEwghVARTbRg53W6d1ILAL_FDWWIBxUboI9_uTKu7HCYuZecU7Uub6MXG5EMWJKUPjVuuHuvkzBs17MdKCRZuwI1fEAU351YQ
  &state=k4y97klszxi
```

A.2.3. 認証エラーレスポンス

A.2.3.1. 認証に失敗した

```
HTTP/1.1 302 Found
Location: https://www.svc.example.net/cb#
  error=access_denied
  &error_description=User%20authentication%20failed.
  &state=k4y97klszxi
```

A.2.3.2. 同意画面でユーザーが拒否した

```
HTTP/1.1 302 Found
Location: https://www.svc.example.net/cb#
  error=access_denied
  &error_description=User%20canceled%20the%20access.
  &state=k4y97klszxi
```

A.2.3.3. scope の値に誤りがある

```
HTTP/1.1 302 Found
Location: https://www.svc.example.net/cb#
  error=invalid_scope
  &error_description=The%20scope%20value%20%22nnn%22%20is%20not%20supported.
  &state=k4y97klszxi
```

A.2.3.4. response_type の値に誤りがある

```
HTTP/1.1 302 Found
Location: https://www.svc.example.net/cb#
  error=unsupported_response_type
  &error_description=The%20response_type%20value%20%22nnn%22%20is%20not%20supported.
  &state=k4y97klszxi
```

A.2.3.5. リクエストパラメータに誤りがある

```
HTTP/1.1 302 Found
Location: https://www.svc.example.net/cb#
  error=invalid_request
  &error_description=%22nnn%22%20is%20required.
  &state=k4y97klszxi
```

A.2.3.6. サーバーサイドの処理でエラーが発生した

```
HTTP/1.1 302 Found
Location: https://www.svc.example.net/cb#
  error=server_error
  &error_description=Error%20occured%20while%20processing%20the%20request.
  &state=k4y97klszxi
```

A.3. OpenID Connect による再認証要求

A.3.1. 認証リクエスト

```
GET /authorize?
  response_type=id_token
  &prompt=login
  &max_age=30
  &login_hint=e1234567
  &client_id=pWBoRam9sG
  &redirect_uri=https%3A%2F%2Fwww.svc.example.net%2Fcb
  &scope=openid
  &state=k4y97klszxi
  &nonce=q8k-upBX4Z_A HTTP/1.1
Host: op.com.example.co.jp
```

A.3.2. 認証成功レスポンス

```
HTTP/1.1 302 Found
Location: https://www.svc.example.net/cb#
  id_token=eyJhbGciOiAi... .eyJpc3MiOiAi... .paxubCBAPw ...
  &state=k4y97klszxi
```

A.3.3. 認証エラーレスポンス

A.3.3.1. 認証に失敗した

```
HTTP/1.1 302 Found
Location: https://www.svc.example.net/cb#
  error=access_denied
  &error_description=User%20authentication%20failed.
  &state=k4y97klszxi
```

A.3.3.2. 同意画面でユーザーが拒否した

```
HTTP/1.1 302 Found
Location: https://www.svc.example.net/cb#
  error=access_denied
  &error_description=User%20canceled%20the%20access.
  &state=k4y97klszxi
```

A.3.3.3. scope の値に誤りがある

```
HTTP/1.1 302 Found
Location: https://www.svc.example.net/cb#
  error=invalid_scope
  &error_description=The%20scope%20value%20%22nnn%22%20is%20not%20supported.
  &state=k4y97klszxi
```

A.3.3.4. response_type の値に誤りがある

```
HTTP/1.1 302 Found
Location: https://www.svc.example.net/cb#
  error=unsupported_response_type
  &error_description=The%20response_type%20value%20%22nnn%22%20is%20not%20supported.
  &state=k4y97klszxi
```

A.3.3.5. リクエストパラメータに誤りがある

```
HTTP/1.1 302 Found
Location: https://www.svc.example.net/cb#
  error=invalid_request
  &error_description=%22nnn%22%20is%20required.
  &state=k4y97klszxi
```

A.3.3.6. サーバサイドの処理でエラーが発生した

```
HTTP/1.1 302 Found
Location: https://www.svc.example.net/cb#
  error=server_error
  &error_description=Error%20occured%20while%20processing%20the%20request.
  &state=k4y97klszxi
```


A.4. OpenID Connect Back-Channel Logout ログアウトトークン

A.4.1. JOSE ヘッダ

```
{
  "alg": "RS256",
  "kid": "iAw5"
}
```

A.4.2. クレーム

```
{
  "iss": "https://op.com.example.co.jp",
  "sub": "e1234567",
  "aud": "pWBoRam9sG",
  "exp": 1435733336,
  "jti": "5ByK",
  "logout_only": true
}
```

A.4.3. ログアウトトークン

```
eyJhbGciOiJSUzI1NiIsImtpZCI6Im1BdzUifQ.eyJpc3MiOiJodHRwczovL29wLmNvbS5leGFtcGxlLmNvLmpwIiwic3ViIjoiaZTEyMzQ1NjciLCJhdwQiOiJwV0JvUmFtOXNHIIiwIZXhwIjoxNDM1NzMzMzY0ZGkiOiI1Qn1LIiwibG9nb3V0X29ubHkiOiJpY0VWV9LzPt-EH3b9mltSem-t4HIq69ygejS0fvVlHFWRzV19ykIBZiADB-soRnRlkjZzCksLDFxNQHC07nKGkqsZkeTGFN52x9tAnV2uvotUOrhrJb5vbbT8YndJqHwSQRqkVec5BHOZ8GFpURTS-9myeKYzgtafJmZab0ZSxRNmkx8YeKVpeH0QRN0gnFi5DxvBva6jTL5m2V9jvgrD2AD8Ix3impYw0X2qe144x1ZaAYaxYr9kwwZY0XDZog4STtsgzRGeiPhXkATyeZQg90FghKBLdVdUvnKjBY9pGdXjGxxSvUXgRQIr9SvYfdyYyP7wpx1uncc0yrq9zesTQJvJnNwx
```

A.5. OpenID Connect Back-Channel Logout による強制ログアウト要求

A.5.1. 強制ログアウト要求

```
POST /bc_logout HTTP/1.1
Host: op.com.example.co.jp
Content-Type: application/x-www-form-urlencoded

logout_token=eyJhbGciOiI... .eyJpc3MiOiI... .zPt-EH3b9m ...
```

A.5.2. 強制ログアウト成功レスポンス

```
HTTP/1.1 200 OK
Cache-Control: no-store
Pragma: no-cache
```

A.5.3. 強制ログアウトエラーレスポンス

A.5.3.1. リクエストにエラーがある

```
HTTP/1.1 400 Bad Request
```

A.5.3.2. ログアウトトークンの検証に失敗した

```
HTTP/1.1 400 Bad Request
```

A.5.3.3. ユーザーとの紐付けに失敗した

```
HTTP/1.1 400 Bad Request
```

A.5.3.4. ログアウト処理が失敗した

```
HTTP/1.1 501 Not Implemented
```

A.5.3.5. 関連サイトのログアウト処理に失敗した

```
HTTP/1.1 504 Gateway Timeout
```

A.6. JWK Set

```
{
  "keys":
  [
    {
      "kty": "RSA",
      "alg": "RS256",
      "kid": "iAw5",
      "n": "26T2mSFqWMIdb4hOBFTQSFtIHD6sZNAphQNR6qzgd-xkY6bPfft
xs0K4178yJ41AWnpMi0sZXyj_dx17L2sJxJQ6R-q8xwMhj-oc9gTLzNK8EF-Fwys
QjT3mcalv9QjYd7-7Tjr9CdytU0dJaF-nxJs0w8Ck519WKRHgLfzEYUeKERnj0
tR8jLfN4DpkcLT1N8bdUgV7nL_d2qpSnuxv83SSPzmuyoq91Xv01TFqHyyK1-fzv
qAoPhLMT-72SdCmev2jNnS2WdDPrRgHvhNsU9XbUoBJswN5yUVFKgprHB2SLntZA
mx9L3YZVLFSGzHRikE2ycZrg1_p0JjRDSwcw",
      "e": "AQAB"
    },
    {
      "kty": "RSA",
      "alg": "RS256",
      "kid": "6CFv",
      "n": "rXdXWSZKW91Ca-sdv9SbCRBcPgnM-OfXuoVg2jDnXHA0cGBi6L
Y0Mys3oePqA1L4dfMFDWQjWzZ1xUW8dcv2TquhJAbQYvrWhIPBksZ2P0tT1Ydme1
rdzwLp4xeeG_C9PA_vipv5kX0BNPltP8ImpsiUmAGQrweUHOWbJ_P_CfJMZ9mev-
QJHmBMM1NgJJg85-QeZcU4Pf6Szxefks9u5tT28CNVYunFxS7IKL0oNk3y8finM8
j06yENTDYMSmot_4iFZORovCSNBN07yH2GsUMsYE7VEvMdxZS1P83mj3ZTMLz49y
xh1NUd_ZXASHIymNPszXyDvmyI-gcbDChjGw",
      "e": "AQAB"
    }
  ]
}
```

付録B.SCIM リクエスト・レスポンス例

B.1. SCIM EIWG 拡張リソース

B.1.1. EIWG 拡張属性

| No | 属性 | 属性名 | サブ属性名 | |
|----|-----------------|---------------------|--------------|--------|
| 1 | 企業 OP でのログイン ID | externalUserName | | |
| 2 | ID トークンの識別子 | idTokenClaims | | |
| 3 | Issuer 識別子 | | issuer | |
| 4 | Subject 識別子 | | subject | |
| 5 | 多言語表現の名前 | | localNames | |
| 6 | 名前の言語 | locale | | |
| 7 | 名前 | formatted | | |
| 8 | 名前 (姓) | familyName | | |
| 9 | 名前 (名) | givenName | | |
| 10 | 表示用の名前 | display | | |
| 11 | 優先フラグ | primary | | |
| 12 | 値の種別 | type | | |
| 13 | 所属組織と役職 | organizationalUnits | | |
| 14 | 組織と所属の言語 | | | locale |
| 15 | 組織コード or 識別子 | | | value |
| 16 | 組織名 | | name | |
| 17 | 表示用の組織名 | | display | |
| 18 | 役職コード or 識別子 | | titleValue | |
| 19 | 役職名 | | titleName | |
| 20 | 表示用の役職名 | | titleDisplay | |
| 21 | 優先フラグ | | primary | |
| 22 | 値の種別 | | type | |

B.1.2. EIWG 拡張スキーマ

```
[
  {
    "id" : "urn:oidfj:params:scim:schemas:extention:enterprisejp:2.0:User",
    "name" : "JapanEnterpriseUser",
    "description" : "Japan Enterprise User",
    "attributes" : [
      {
        "name" : "extenalUserName",
        "type" : "string",
        "multiValued" : false,
        "description" : "The username the User uses to login to OP.",
        "required" : true,
        "caseExact" : true,
        "mutability" : "readWrite",
        "returned" : "default",
        "uniqueness" : "none"
      },
      {
        "name" : "idTokenClaims",
        "type" : "complex",
        "multiValued" : false,
        "description" : "The values of ID Token claims which used to identify
authenticated user.",
        "required" : true,
        "subAttributes" : [
          {
            "name" : "issuer",
            "type" : "string",
            "multiValued" : false,
            "description" : "The value of ID Token iss claim.",
            "required" : true,
            "caseExact" : true,
            "mutability" : "readWrite",
            "returned" : "default",
            "uniqueness" : "none"
          },
          {
            "name" : "subject",
            "type" : "string",
            "multiValued" : false,
            "description" : "The value of ID Token sub claim.",
            "required" : true,
            "caseExact" : true,
            "mutability" : "readWrite",
            "returned" : "default",
            "uniqueness" : "none"
          }
        ]
      },
      {
        "name" : "subject",
        "type" : "string",
        "multiValued" : false,
        "description" : "The value of ID Token sub claim.",
        "required" : true,
        "caseExact" : true,
        "mutability" : "readWrite",
        "returned" : "default",
        "uniqueness" : "none"
      }
    ],
    "mutability" : "readWrite",
    "returned" : "default",
    "uniqueness" : "none"
  }
]
```

```
    },
    {
      "name" : "localNames",
      "type" : "complex",
      "multiValued" : true,
      "description" : "The components of the user's real name in various
languages.",
      "required" : false,
      "subAttributes" : [
        {
          "name" : "locale",
          "type" : "string",
          "multiValued" : false,
          "description" : "A label indicating the attribute's language.",
          "required" : false,
          "caseExact" : false,
          "mutability" : "readWrite",
          "returned" : "default",
          "uniqueness" : "none"
        },
        {
          "name" : "formatted",
          "type" : "string",
          "multiValued" : false,
          "description" : "The full name of the User.",
          "required" : false,
          "caseExact" : false,
          "mutability" : "readWrite",
          "returned" : "default",
          "uniqueness" : "none"
        },
        {
          "name" : "familyName",
          "type" : "string",
          "multiValued" : false,
          "description" : "The family name of the User.",
          "required" : false,
          "caseExact" : false,
          "mutability" : "readWrite",
          "returned" : "default",
          "uniqueness" : "none"
        },
        {
          "name" : "givenName",
          "type" : "string",
          "multiValued" : false,
          "description" : "The given name of the User.",
          "required" : false,
          "caseExact" : false,
          "mutability" : "readWrite",
          "returned" : "default",
          "uniqueness" : "none"
        }
      ],
    },
  ],
}
```

```
{
  "name" : "display",
  "type" : "string",
  "multiValued" : false,
  "description" : "The name of the User, suitable for display to end-
users.",
  "required" : false,
  "caseExact" : false,
  "mutability" : "readWrite",
  "returned" : "default",
  "uniqueness" : "none"
},
{
  "name" : "primary",
  "type" : "boolean",
  "multiValued" : false,
  "description" : "A Boolean value indicating the 'primary' or preferred
attribute value for this attribute.",
  "required" : false,
  "mutability" : "readWrite",
  "returned" : "default"
},
{
  "name" : "type",
  "type" : "string",
  "multiValued" : false,
  "description" : "A label indicating the attribute's function.",
  "required" : false,
  "caseExact" : false,
  "mutability" : "readWrite",
  "returned" : "default",
  "uniqueness" : "none"
}
],
"mutability" : "readWrite",
"returned" : "default",
"uniqueness" : "none"
},
{
  "name" : "organizationalUnits",
  "type" : "complex",
  "multiValued" : true,
  "description" : "The name of department and title of the User in various
languages.",
  "required" : false,
  "subAttributes" : [
    {
      "name" : "locale",
      "type" : "string",
      "multiValued" : false,
      "description" : "A label indicating the attribute's language.",
      "required" : false,
      "caseExact" : false,
```

```
    "mutability" : "readWrite",
    "returned" : "default",
    "uniqueness" : "none"
  },
  {
    "name" : "value",
    "type" : "string",
    "multiValued" : false,
    "description" : "The identifier of a department.",
    "required" : false,
    "caseExact" : false,
    "mutability" : "readWrite",
    "returned" : "default",
    "uniqueness" : "none"
  },
  {
    "name" : "name",
    "type" : "string",
    "multiValued" : false,
    "description" : "The name of a department.",
    "required" : false,
    "caseExact" : false,
    "mutability" : "readWrite",
    "returned" : "default",
    "uniqueness" : "none"
  },
  {
    "name" : "display",
    "type" : "string",
    "multiValued" : false,
    "description" : "The name of a department, suitable for display to
end-users.",
    "required" : false,
    "caseExact" : false,
    "mutability" : "readWrite",
    "returned" : "default",
    "uniqueness" : "none"
  },
  {
    "name" : "titleValue",
    "type" : "string",
    "multiValued" : false,
    "description" : "The identifier of a title.",
    "required" : false,
    "caseExact" : false,
    "mutability" : "readWrite",
    "returned" : "default",
    "uniqueness" : "none"
  },
  {
    "name" : "titleName",
    "type" : "string",
    "multiValued" : false,
```



```
    "description" : "The name of a title.",
    "required" : false,
    "caseExact" : false,
    "mutability" : "readWrite",
    "returned" : "default",
    "uniqueness" : "none"
  },
  {
    "name" : "titleDisplay",
    "type" : "string",
    "multiValued" : false,
    "description" : "The name of a title, suitable for display to end-
users.",
    "required" : false,
    "caseExact" : false,
    "mutability" : "readWrite",
    "returned" : "default",
    "uniqueness" : "none"
  },
  {
    "name" : "primary",
    "type" : "boolean",
    "multiValued" : false,
    "description" : "A Boolean value indicating the 'primary' or preferred
attribute value for this attribute.",
    "required" : false,
    "mutability" : "readWrite",
    "returned" : "default"
  },
  {
    "name" : "type",
    "type" : "string",
    "multiValued" : false,
    "description" : "A label indicating the attribute's function.",
    "required" : false,
    "caseExact" : false,
    "mutability" : "readWrite",
    "returned" : "default",
    "uniqueness" : "none"
  }
],
"mutability" : "readWrite",
"returned" : "default",
"uniqueness" : "none"
}
],
"meta" : {
  "resourceType" : "Schema",
  "location" : "/v2/Schemas/urn:oidfj:params:scim:schemas:extention:enterprise
jp:2.0:User"
}
}
]
```

B.2. SCIM ユーザーの作成

B.2.1. ユーザー作成リクエスト

B.2.1.1. 作成するユーザーの属性

| No | 属性 | 例 | SCIM 属性名 | サブ属性名 |
|----|-----------------|-------------------------------|------------------|------------|
| 1 | ID 管理サーバー上の識別子 | e1234567 | externalId | |
| 2 | サービス利用時のユーザー名 | taro.nippon@com.example.co.jp | userName | |
| 3 | 企業 OP でのログイン ID | e1234567 | externalUserName | |
| 4 | ID トークンの識別子 | — | idTokenClaims | |
| 5 | Issuer 識別子 | https://op.com.example.co.jp | | issuer |
| 6 | Subject 識別子 | e1234567 | | subject |
| 7 | 従業員番号 | e1234567 | employeeNumber | |
| 8 | 名前 (母国語) | — | name | |
| 9 | 名前 | 日本 太郎 | | formatted |
| 10 | 名前 (姓) | 日本 | | familyName |
| 11 | 名前 (名) | 太郎 | | givenName |
| 12 | 表示用の名前 (母国語) | 日本 太郎 | displayName | |
| 13 | 所属 (主務・母国語) | 営業部営業 1 課 | department | |
| 14 | 役職 (主務・母国語) | 課長 | title | |
| 15 | 地域 | ja-JP | locale | |
| 16 | メールアドレス | — | emails | |
| 17 | メールアドレス | taro.nippon@com.example.co.jp | | value |
| 18 | 優先フラグ | true | | primary |
| 19 | 電話番号 (会社) | — | phoneNumbers | |
| 20 | 値の種別 | work | | type |
| 21 | 電話番号 | 03-1234-5678 | | value |
| 22 | 優先フラグ | true | | primary |
| 23 | 電話番号 (携帯電話) | — | phoneNumbers | |
| 25 | 値の種別 | mobile | | type |

| | | | | |
|----|-----------------|---------------|--------------|------------|
| 24 | 電話番号 | 090-1234-5678 | | value |
| 26 | 優先フラグ | false | | primary |
| 27 | 電話番号 (内線) | — | phoneNumbers | |
| 29 | 値の種別 | extention | | type |
| 28 | 電話番号 | 9999 | | value |
| 30 | 優先フラグ | false | | primary |
| 31 | 有効フラグ | true | | active |
| 32 | 多言語表現の名前 (漢字) | — | localNames | |
| 33 | 名前の言語 | ja-JP | | locale |
| 34 | 名前 | 日本 太郎 | | formatted |
| 35 | 名前 (姓) | 日本 | | familyName |
| 36 | 名前 (名) | 太郎 | | givenName |
| 37 | 表示用の名前 | 日本 太郎 | | display |
| 38 | 優先フラグ | true | | primary |
| 39 | 値の種別 | ja-JP | | type |
| 40 | 多言語表現の名前 (よみがな) | — | | localNames |
| 41 | 名前の言語 | ja-Hira-JP | locale | |
| 42 | 名前 | にっぽん たらう | formatted | |
| 43 | 名前 (姓) | にっぽん | familyName | |
| 44 | 名前 (名) | たらう | givenName | |
| 45 | 表示用の名前 | にっぽん たらう | display | |
| 46 | 優先フラグ | false | primary | |
| 47 | 値の種別 | ja-JP | type | |
| 48 | 多言語表現の名前 (ローマ字) | — | localNames | |
| 49 | 名前の言語 | en-US | | locale |
| 50 | 名前 | Taro Nippon | | formatted |
| 51 | 名前 (姓) | Nippon | | familyName |
| 52 | 名前 (名) | Taro | | givenName |
| 53 | 表示用の名前 | Taro Nippon | | display |
| 54 | 優先フラグ | false | | primary |
| 55 | 値の種別 | ja-JP | | type |

| | | | | |
|----|------------------|---------------------------------------|---------------------|---------------------|
| 56 | 所属組織と役職（主務・漢字） | — | organizationalUnits | |
| 57 | 組織と役職の言語 | ja-JP | | locale |
| 58 | 組織コード or 識別子 | 10010000 | | value |
| 59 | 組織名 | 営業 1 課 | | name |
| 60 | 表示用の組織名 | 営業部営業 1 課 | | display |
| 61 | 役職コード or 識別子 | 5000 | | titleValue |
| 62 | 役職名 | 課長 | | titleName |
| 63 | 表示用の役職名 | 課長 | | titleDisplay |
| 64 | 優先フラグ | true | | primary |
| 65 | 値の種別 | ja-JP | | type |
| 66 | 所属組織と役職（主務・ローマ字） | — | | organizationalUnits |
| 67 | 組織と役職の言語 | en-US | locale | |
| 68 | 組織コード or 識別子 | 10010000 | value | |
| 69 | 組織名 | Sales Group I | name | |
| 70 | 表示用の組織名 | Sales Group I, Sales Department | display | |
| 71 | 役職コード or 識別子 | 5000 | titleValue | |
| 72 | 役職名 | Manager | titleName | |
| 73 | 表示用の役職名 | Manager | titleDisplay | |
| 74 | 優先フラグ | false | primary | |
| 75 | 値の種別 | en-US | type | |

B.2.1.2. SCIM リクエスト

```
POST /Users HTTP/1.1
Host: scim.svc.example.net
Accept: application/scim+json
Content-Type: application/scim+json
Authorization: Basic Yzc2NTQzMjE6Z3l0OUUpSN3pUWDRteXd0NHFiMW5ZdUhpTTA3WFIE0Q1U=
Content-Length: ...

{
  "schemas": [
    "urn:ietf:params:scim:schemas:core:2.0:User",
    "urn:ietf:params:scim:schemas:extension:enterprise:2.0:User",
    "urn:oidfj:params:scim:schemas:extention:enterprisejp:2.0:User"
  ],
  "userName": "taro.nippon@com.example.co.jp",
  "externalId": "e1234567",
  "name": {
    "formatted": "日本 太郎",
    "familyName": "日本",
    "givenName": "太郎"
  },
  "displayName": "日本 太郎",
  "department": "営業部営業1課",
  "title": "課長",
  "locale": "ja-JP",
  "emails": [
    {
      "value": "taro.nippon@com.example.co.jp",
      "primary": true
    }
  ],
  "phoneNumbers": [
    {
      "type": "work",
      "value": "03-1234-5678",
      "primary": true
    },
    {
      "type": "mobile",
      "value": "090-1234-5678",
      "primary": false
    },
    {
      "type": "extention",
      "value": "9999",
      "primary": false
    }
  ],
  "active": true,
  "urn:ietf:params:scim:schemas:extension:enterprise:2.0:User": {
    "employeeNumber": "e1234567"
  }
}
```

```
},
"urn:oid:urn:ietf:params:scim:schemas:extension:enterprise:2.0:User": {
  "externalUserName": "e1234567",
  "idTokenClaims": {
    "issuer": "https://op.com.example.co.jp",
    "subject": "e1234567"
  },
  "localNames": [
    {
      "locale": "ja-JP",
      "formatted": "日本 太郎",
      "familyName": "日本",
      "givenName": "太郎",
      "display": "日本 太郎",
      "primary": true,
      "type": "ja-JP"
    },
    {
      "locale": "ja-Hira-JP",
      "formatted": "にっぽん たろう",
      "familyName": "にっぽん",
      "givenName": "たろう",
      "display": "にっぽん たろう",
      "primary": false,
      "type": "ja-Hira-JP"
    },
    {
      "locale": "en-US",
      "formatted": "Taro Nippon",
      "familyName": "Nippon",
      "givenName": "Taro",
      "display": "Taro Nippon",
      "primary": false,
      "type": "en-US"
    }
  ],
  "organizationalUnits": [
    {
      "locale": "ja-JP",
      "value": "10010000",
      "name": "営業1課",
      "display": "営業部営業1課",
      "titleValue": "5000",
      "titleName": "課長",
      "titleDisplay": "課長",
      "primary": true,
      "type": "ja-JP"
    },
    {
      "locale": "en-US",
      "value": "10010000",
      "name": "Sales Group I",
```

```

    "display": "Sales Group I, Sales Department",
    "titleValue": "5000",
    "titleName": "Manager",
    "titleDisplay": "Manager",
    "primary": false,
    "type": "en-US"
  }
]
}
}

```

B.2.2. ユーザー作成成功レスポンス

B.2.2.1. サーバーが付与する属性

| No | 属性 | 値 |
|----|-------|--------------------------------------|
| 1 | ID | a2e492da-e2ed-4d90-a186-6fc01b56b8d9 |
| 2 | バージョン | 4124bc0a9335c27f086f24ba207a4912 |
| 3 | 作成日 | 2015-04-01T01:23:45.678Z |
| 4 | 更新日 | 2015-04-01T01:23:45.678Z |

B.2.2.2. SCIM レスポンス

```

HTTP/1.1 201 Created
Content-Type: application/scim+json
Location: https://scim.svc.example.net/v2/User/a2e492da-e2ed-4d90-a186-6fc01b56b8d9
ETag: W/"4124bc0a9335c27f086f24ba207a4912"

{
  "schemas": [
    "urn:ietf:params:scim:schemas:core:2.0:User",
    "urn:ietf:params:scim:schemas:extension:enterprise:2.0:User",
    "urn:oidfj:params:scim:schemas:extention:enterprisejp:2.0:User"
  ],
  "id": "a2e492da-e2ed-4d90-a186-6fc01b56b8d9",
  "userName": "taro.nippon@com.example.co.jp",
  "externalId": "e1234567",
  "meta": {
    "resourceType": "User",
    "created": "2015-04-01T01:23:45.678Z",
    "lastModified": "2015-04-01T01:23:45.678Z",
    "location": "https://scim.svc.example.net/v2/User/a2e492da-e2ed-4d90-a186-6fc01b56b8d9",
    "version": "W/\\"4124bc0a9335c27f086f24ba207a4912\""
  },
  "name": {
    "formatted": "日本 太郎",

```

```
"familyName": "日本",
"givenName": "太郎"
},
"displayName": "日本 太郎",
"department": "営業部営業1課",
"title": "課長",
"locale": "ja-JP",
"emails": [
  {
    "value": "taro.nippon@com.example.co.jp",
    "primary": true
  }
],
"phoneNumbers": [
  {
    "type": "work",
    "value": "03-1234-5678",
    "primary": true
  },
  {
    "type": "mobile",
    "value": "090-1234-5678",
    "primary": false
  },
  {
    "type": "extention",
    "value": "9999",
    "primary": false
  }
],
"active": true,
"urn:iETF:params:scim:schemas:extension:enterprise:2.0:User": {
  "employeeNumber": "e1234567"
},
"urn:oidfj:params:scim:schemas:extention:enterprisejp:2.0:User": {
  "externalUserName": "e1234567",
  "idTokenClaims": {
    "issuer": "https://op.com.example.co.jp",
    "subject": "e1234567"
  }
},
"localNames": [
  {
    "locale": "ja-JP",
    "formatted": "日本 太郎",
    "familyName": "日本",
    "givenName": "太郎",
    "display": "日本 太郎",
    "primary": true,
    "type": "ja-JP"
  },
  {
    "locale": "ja-Hira-JP",
```



```
"formatted": "にっぽん たろう",
"familyName": "にっぽん",
"givenName": "たろう",
"display": "にっぽん たろう",
"primary": false,
"type": "ja-Hira-JP"
},
{
  "locale": "en-US",
  "formatted": "Taro Nippon",
  "familyName": "Nippon",
  "givenName": "Taro",
  "display": "Taro Nippon",
  "primary": false,
  "type": "en-US"
}
],
"organizationalUnits": [
  {
    "locale": "ja-JP",
    "value": "10010000",
    "name": "営業 1 課",
    "display": "営業部営業 1 課",
    "titleValue": "5000",
    "titleName": "課長",
    "titleDisplay": "課長",
    "primary": true,
    "type": "ja-JP"
  },
  {
    "locale": "en-US",
    "value": "10010000",
    "name": "Sales Group I",
    "display": "Sales Group I, Sales Department",
    "titleValue": "5000",
    "titleName": "Manager",
    "titleDisplay": "Manager",
    "primary": false,
    "type": "en-US"
  }
]
}
```

B.2.3. ユーザー作成エラーレスポンス

エラーが発生した場合のレスポンスを例示する。利用企業の ID 管理サーバーにエラーの内容を伝達するために、SCIM サーバーは "detail" 属性にエラーの詳細を記述して応答することが望ましい。

B.2.3.1. リクエストメッセージに誤りがある

```
HTTP/1.1 400 Bad Request
Content-Type: application/scim+json

{
  "schemas": ["urn:ietf:params:scim:api:messages:2.0:Error"],
  "detail": "SCIM サーバーがエラーの詳細を記入する",
  "scimType": "invalidSyntax",
  "status": "400"
}
```

B.2.3.2. ユーザー属性値に誤りがある

```
HTTP/1.1 400 Bad Request
Content-Type: application/scim+json

{
  "schemas": ["urn:ietf:params:scim:api:messages:2.0:Error"],
  "detail": "SCIM サーバーがエラーの詳細を記入する",
  "scimType": "invalidValue",
  "status": "400"
}
```

B.2.3.3. ユーザー属性値が重複している

```
HTTP/1.1 400 Bad Request
Content-Type: application/scim+json

{
  "schemas": ["urn:ietf:params:scim:api:messages:2.0:Error"],
  "detail": "SCIM サーバーがエラーの詳細を記入する",
  "scimType": "uniqueness",
  "status": "400"
}
```

B.2.3.4. ユーザー作成処理中にエラーが発生した

```
HTTP/1.1 500 Internal Server Error
Content-Type: application/scim+json

{
  "schemas": ["urn:ietf:params:scim:api:messages:2.0:Error"],
  "detail": "SCIM サーバーがエラーの詳細を記入する",
  "status": "500"
}
```

B.2.3.5. 認証に失敗した

```
HTTP/1.1 401 Unauthorized
Content-Type: application/scim+json

{
  "schemas": ["urn:ietf:params:scim:api:messages:2.0:Error"],
  "detail": "SCIM サーバーがエラーの詳細を記入する",
  "status": "401"
}
```

B.2.3.6. リソースへのアクセス・操作権限がない

```
HTTP/1.1 403 Forbidden
Content-Type: application/scim+json

{
  "schemas": ["urn:ietf:params:scim:api:messages:2.0:Error"],
  "detail": "SCIM サーバーがエラーの詳細を記入する",
  "status": "403"
}
```

B.3. SCIM ユーザーの検索

B.3.1. ユーザー検索リクエスト

B.3.1.1. ユーザー検索条件

| No | 属性 | 値 |
|----|------------|----------|
| 1 | externalId | e1234567 |

B.3.1.2. SCIM リクエスト

```
POST /.search HTTP/1.1
Host: scim.svc.example.net
Accept: application/scim+json
Content-Type: application/scim+json
Authorization: Basic Yzc2NTQzMjE6Z3l00UpSN3pUWDRteXd0NHFiMW5ZdUhpTTA3WF10Q1U=
Content-Length: ...

{
  "schemas": ["urn:ietf:params:scim:api:messages:2.0:SearchRequest"],
  "attributes": ["externalId", "meta"],
  "filter": "externalId eq \"e1234567\""
}
```

B.3.2. ユーザー検索成功レスポンス

B.3.2.1. SCIM レスポンス

```
HTTP/1.1 200 OK
Content-Type: application/scim+json

{
  "schemas": [
    "urn:ietf:params:scim:api:messages:2.0:ListResponse"
  ],
  "totalResults": 1,
  "Resources": [
    {
      "id": "a2e492da-e2ed-4d90-a186-6fc01b56b8d9",
      "externalId": "e1234567",
      "meta": {
        "resourceType": "User",
        "created": "2015-04-01T01:23:45.678Z",
        "lastModified": "2015-04-01T01:23:45.678Z",
        "location": "https://scim.svc.example.net/v2/User/a2e492da-e2ed-4d90-a186-6fc01b56b8d9",
        "version": "W/\"4124bc0a9335c27f086f24ba207a4912\""
      }
    }
  ]
}
```

B.3.3. ユーザー検索エラーレスポンス

エラーが発生した場合のレスポンスを例示する。利用企業の ID 管理サーバーにエラーの内容を伝達するために、SCIM サーバーは "detail" 属性にエラーの詳細を記述して応答することが望ましい。

B.3.3.1. ユーザーが見つからない

ユーザーリソース識別子 (id) 取得のための検索のため、検索結果が 1 件ではないレスポンスはエラーレスポンスとなる。

```
HTTP/1.1 200 OK
Content-Type: application/scim+json

{
  "schemas": [
    "urn:ietf:params:scim:api:messages:2.0:ListResponse"
  ],
  "totalResults": 0,
  "Resources": []
}
```

B.3.3.2. リクエストメッセージに誤りがある

```
HTTP/1.1 400 Bad Request
Content-Type: application/scim+json

{
  "schemas": ["urn:ietf:params:scim:api:messages:2.0:Error"],
  "detail": "SCIM サーバーがエラーの詳細を記入する",
  "scimType": "invalidSyntax",
  "status": "400"
}
```

B.3.3.3. リクエストのフィルター設定に誤りがある

```
HTTP/1.1 400 Bad Request
Content-Type: application/scim+json

{
  "schemas": ["urn:ietf:params:scim:api:messages:2.0:Error"],
  "detail": "SCIM サーバーがエラーの詳細を記入する",
  "scimType": "invalidFilter",
  "status": "400"
}
```

B.3.3.4. ユーザー検索処理中にエラーが発生した

```
HTTP/1.1 500 Internal Server Error
Content-Type: application/scim+json

{
  "schemas": ["urn:ietf:params:scim:api:messages:2.0:Error"],
  "detail": "SCIM サーバーがエラーの詳細を記入する",
  "status": "500"
}
```

B.3.3.5. 認証に失敗した

```
HTTP/1.1 401 Unauthorized
Content-Type: application/scim+json

{
  "schemas": ["urn:ietf:params:scim:api:messages:2.0:Error"],
  "detail": "SCIM サーバーがエラーの詳細を記入する",
  "status": "401"
}
```

B.3.3.6. リソースへのアクセス・操作権限がない

```
HTTP/1.1 403 Forbidden
Content-Type: application/scim+json

{
  "schemas": ["urn:ietf:params:scim:api:messages:2.0:Error"],
  "detail": "SCIM サーバーがエラーの詳細を記入する",
  "status": "403"
}
```

B.4. SCIM ユーザーの更新

SCIM でのユーザーの更新は、(1) 更新対象ユーザーの SCIM サーバー上でのユーザーリソース識別子 (id) を取得し、(2) ユーザーの全属性を置き換える、手順で実行する。

B.4.1. ユーザーリソース識別子取得

B.3. SCIM ユーザーの問合せに記載の方法で、更新対象ユーザーの SCIM サーバー上でのユーザーリソース識別子 (id) を取得する。

問合せの結果、次のユーザーリソースが取得できたものとして、以降の例を記載する。

| No | 属性 | 値 |
|----|--------------|--------------------------------------|
| 1 | id | a2e492da-e2ed-4d90-a186-6fc01b56b8d9 |
| 2 | externalId | e1234567 |
| 3 | meta.version | W/"4124bc0a9335c27f086f24ba207a4912" |

B.4.2. ユーザー更新リクエスト

B.4.2.1. ユーザー更新条件

| No | 属性 | 変更前 | 変更後 |
|----|----|-----------|-----------|
| 1 | 所属 | 営業部営業 1 課 | 営業部営業 2 課 |

B.4.2.2. SCIM リクエスト

```
PUT /Users/a2e492da-e2ed-4d90-a186-6fc01b56b8d9 HTTP/1.1
Host: scim.svc.example.net
Accept: application/scim+json
Content-Type: application/scim+json
Authorization: Basic Yzc2NTQzMjE6Z3l0UUpSN3pUWDRteXd0NHFiMW5ZdUhpTTA3WF10Q1U=
Content-Length: ...
If-Match: W/"4124bc0a9335c27f086f24ba207a4912"

{
  "schemas": [
    "urn:ietf:params:scim:schemas:core:2.0:User",
    "urn:ietf:params:scim:schemas:extension:enterprise:2.0:User",
    "urn:oidfj:params:scim:schemas:extention:enterprisejp:2.0:User"
  ],
  "id": "a2e492da-e2ed-4d90-a186-6fc01b56b8d9",
  "userName": "taro.nippon@com.example.co.jp",
  "externalId": "e1234567",
  "name": {
    "formatted": "日本 太郎",
    "familyName": "日本",
    "givenName": "太郎"
  },
  "displayName": "日本 太郎",
  "department": "営業部営業 2 課",
  "title": "課長",
  "locale": "ja-JP",
  "emails": [
    {
      "value": "taro.nippon@com.example.co.jp",
      "primary": true
    }
  ]
}
```

```
}
],
"phoneNumbers": [
  {
    "type": "work",
    "value": "03-1234-5678",
    "primary": true
  },
  {
    "type": "mobile",
    "value": "090-1234-5678",
    "primary": false
  },
  {
    "type": "extention",
    "value": "9999",
    "primary": false
  }
],
"active": true,
"urn:ietf:params:scim:schemas:extension:enterprise:2.0:User": {
  "employeeNumber": "e1234567"
},
"urn:oidfj:params:scim:schemas:extention:enterprisejp:2.0:User": {
  "externalUserName": "e1234567",
  "idTokenClaims": {
    "issuer": "https://op.com.example.co.jp",
    "subject": "e1234567"
  }
},
"localNames": [
  {
    "locale": "ja-JP",
    "formatted": "日本 太郎",
    "familyName": "日本",
    "givenName": "太郎",
    "display": "日本 太郎",
    "primary": true,
    "type": "ja-JP"
  },
  {
    "locale": "ja-Hira-JP",
    "formatted": "にっぽん たろう",
    "familyName": "にっぽん",
    "givenName": "たろう",
    "display": "にっぽん たろう",
    "primary": false,
    "type": "ja-Hira-JP"
  },
  {
    "locale": "en-US",
    "formatted": "Taro Nippon",
    "familyName": "Nippon",
```



```

    "givenName": "Taro",
    "display": "Taro Nippon",
    "primary": false,
    "type": "en-US"
  }
],
"organizationalUnits": [
  {
    "locale": "ja-JP",
    "value": "10020000",
    "name": "営業 2 課",
    "display": "営業部営業 2 課",
    "titleValue": "5000",
    "titleName": "課長",
    "titleDisplay": "課長",
    "primary": true,
    "type": "ja-JP"
  },
  {
    "locale": "en-US",
    "value": "10020000",
    "name": "Sales Group II",
    "display": "Sales Group II, Sales Department",
    "titleValue": "5000",
    "titleName": "Manager",
    "titleDisplay": "Manager",
    "primary": false,
    "type": "en-US"
  }
]
}
}

```

B.4.3. ユーザー更新成功レスポンス

B.4.3.1. サーバが更新する属性

| No | 属性 | 値 |
|----|-------|----------------------------------|
| 1 | バージョン | 21ad0bd836b90d08f4cf640b4c298e7c |
| 2 | 更新日 | 2015-07-01T01:02:03.004Z |

B.4.3.2. SCIM レスポンス

```

HTTP/1.1 200 OK
Content-Type: application/scim+json
Location: https://scim.svc.example.net/v2/User/a2e492da-e2ed-4d90-a186-6fc01b56b8d9
ETag: W/"21ad0bd836b90d08f4cf640b4c298e7c"

```

```
{
  "schemas": [
    "urn:ietf:params:scim:schemas:core:2.0:User",
    "urn:ietf:params:scim:schemas:extension:enterprise:2.0:User",
    "urn:oidfj:params:scim:schemas:extention:enterprisejp:2.0:User"
  ],
  "id": "a2e492da-e2ed-4d90-a186-6fc01b56b8d9",
  "userName": "taro.nippon@com.example.co.jp",
  "externalId": "e1234567",
  "meta": {
    "resourceType": "User",
    "created": "2015-04-01T01:23:45.678Z",
    "lastModified": "2015-07-01T01:02:03.004Z",
    "location": "https://scim.svc.example.net/v2/User/a2e492da-e2ed-4d90-a186-6fc01b56b8d9",
    "version": "W/\\"21ad0bd836b90d08f4cf640b4c298e7c\\""
  },
  "name": {
    "formatted": "日本 太郎",
    "familyName": "日本",
    "givenName": "太郎"
  },
  "displayName": "日本 太郎",
  "department": "営業部営業 2 課",
  "title": "課長",
  "locale": "ja-JP",
  "emails": [
    {
      "value": "taro.nippon@com.example.co.jp",
      "primary": true
    }
  ],
  "phoneNumbers": [
    {
      "type": "work",
      "value": "03-1234-5678",
      "primary": true
    },
    {
      "type": "mobile",
      "value": "090-1234-5678",
      "primary": false
    },
    {
      "type": "extention",
      "value": "9999",
      "primary": false
    }
  ],
  "active": true,
  "urn:ietf:params:scim:schemas:extension:enterprise:2.0:User": {
    "employeeNumber": "e1234567"
  },
}
```

```
"urn:oidfj:params:scim:schemas:extention:enterprisejp:2.0:User": {
  "externalUserName": "e1234567",
  "idTokenClaims": {
    "issuer": "https://op.com.example.co.jp",
    "subject": "e1234567"
  },
  "localNames": [
    {
      "locale": "ja-JP",
      "formatted": "日本 太郎",
      "familyName": "日本",
      "givenName": "太郎",
      "display": "日本 太郎",
      "primary": true,
      "type": "ja-JP"
    },
    {
      "locale": "ja-Hira-JP",
      "formatted": "にっぽん たろう",
      "familyName": "にっぽん",
      "givenName": "たろう",
      "display": "にっぽん たろう",
      "primary": false,
      "type": "ja-Hira-JP"
    },
    {
      "locale": "en-US",
      "formatted": "Taro Nippon",
      "familyName": "Nippon",
      "givenName": "Taro",
      "display": "Taro Nippon",
      "primary": false,
      "type": "en-US"
    }
  ],
  "organizationalUnits": [
    {
      "locale": "ja-JP",
      "value": "10020000",
      "name": "営業 2 課",
      "display": "営業部営業 2 課",
      "titleValue": "5000",
      "titleName": "課長",
      "titleDisplay": "課長",
      "primary": true,
      "type": "ja-JP"
    },
    {
      "locale": "en-US",
      "value": "10020000",
      "name": "Sales Group II",
      "display": "Sales Group II, Sales Department",
```

```
"titleValue": "5000",
"titleName": "Manager",
"titleDisplay": "Manager",
"primary": false,
"type": "en-US"
}
]
}
}
```

B.4.4. ユーザー更新エラーレスポンス

エラーが発生した場合のレスポンスを例示する。利用企業の ID 管理サーバーにエラーの内容を伝達するために、SCIM サーバーは "detail" 属性にエラーの詳細を記述して応答することが望ましい。

B.4.4.1. ユーザーリソースが見つからない

```
HTTP/1.1 404 Not Found
Content-Type: application/scim+json

{
  "schemas": ["urn:ietf:params:scim:api:messages:2.0:Error"],
  "detail": "SCIM サーバーがエラーの詳細を記入する",
  "status": "404"
}
```

B.4.4.2. リクエストメッセージに誤りがある

```
HTTP/1.1 400 Bad Request
Content-Type: application/scim+json

{
  "schemas": ["urn:ietf:params:scim:api:messages:2.0:Error"],
  "detail": "SCIM サーバーがエラーの詳細を記入する",
  "scimType": "invalidSyntax",
  "status": "400"
}
```

B.4.4.3. ユーザー属性値に誤りがある

```
HTTP/1.1 400 Bad Request
Content-Type: application/scim+json

{
  "schemas": ["urn:ietf:params:scim:api:messages:2.0:Error"],
  "detail": "SCIM サーバーがエラーの詳細を記入する",
  "scimType": "invalidValue",
  "status": "400"
}
```

B.4.4.4. ユーザー属性値が重複している

```
HTTP/1.1 400 Bad Request
Content-Type: application/scim+json

{
  "schemas": ["urn:ietf:params:scim:api:messages:2.0:Error"],
  "detail": "SCIM サーバーがエラーの詳細を記入する",
  "scimType": "uniqueness",
  "status": "400"
}
```

B.4.4.5. ユーザーの更新要求が競合した

```
HTTP/1.1 409 Conflict
Content-Type: application/scim+json

{
  "schemas": ["urn:ietf:params:scim:api:messages:2.0:Error"],
  "detail": "SCIM サーバーがエラーの詳細を記入する",
  "status": "409"
}
```

B.4.4.6. ユーザー更新処理中にエラーが発生した

```
HTTP/1.1 500 Internal Server Error
Content-Type: application/scim+json

{
  "schemas": ["urn:ietf:params:scim:api:messages:2.0:Error"],
  "detail": "SCIM サーバーがエラーの詳細を記入する",
  "status": "500"
}
```

B.4.4.7. 認証に失敗した

```
HTTP/1.1 401 Unauthorized
Content-Type: application/scim+json

{
  "schemas": ["urn:ietf:params:scim:api:messages:2.0:Error"],
  "detail": "SCIM サーバーがエラーの詳細を記入する",
  "status": "401"
}
```

B.4.4.8. リソースへのアクセス・操作権限がない

```
HTTP/1.1 403 Forbidden
Content-Type: application/scim+json

{
  "schemas": ["urn:ietf:params:scim:api:messages:2.0:Error"],
  "detail": "SCIM サーバーがエラーの詳細を記入する",
  "status": "403"
}
```

B.5. SCIM ユーザーの削除

SCIM でのユーザーの削除は、(1) 削除対象ユーザーの SCIM サーバー上でのユーザーリソース識別子 (id) を取得し、(2) ユーザーを削除する、手順で実行する。

B.5.1. ユーザーリソース識別子取得

B.3. SCIM ユーザーの問合せに記載の方法で、更新対象ユーザーの SCIM サーバー上でのユーザーリソース識別子 (id) を取得する。

問合せの結果、次のユーザーリソースが取得できたものとして、以降の例を記載する。

| No | 属性 | 値 |
|----|--------------|--------------------------------------|
| 1 | id | a2e492da-e2ed-4d90-a186-6fc01b56b8d9 |
| 2 | externalId | e1234567 |
| 3 | meta.version | W/"4124bc0a9335c27f086f24ba207a4912" |

B.5.2. ユーザー削除リクエスト

B.5.2.1. SCIM リクエスト

```
DELETE /Users/a2e492da-e2ed-4d90-a186-6fc01b56b8d9 HTTP/1.1
Host: scim.svc.example.net
Authorization: Basic Yzc2NTQzMjE6Z3100UpSN3pUWDRteXd0NHFiMW5ZdUhpTTA3WF10Q1U=
If-Match: W/"4124bc0a9335c27f086f24ba207a4912"
```

B.5.3. ユーザー削除成功レスポンス

B.5.3.1. SCIM レスポンス

```
HTTP/1.1 204 No Content
```

B.5.4. ユーザー削除エラーレスポンス

エラーが発生した場合のレスポンスを例示する。利用企業の ID 管理サーバーにエラーの内容を伝達するために、SCIM サーバーは "detail" 属性にエラーの詳細を記述して応答することが望ましい。

B.5.4.1. ユーザーリソースが見つからない

```
HTTP/1.1 404 Not Found
Content-Type: application/scim+json

{
  "schemas": ["urn:ietf:params:scim:api:messages:2.0:Error"],
  "detail": "SCIM サーバーがエラーの詳細を記入する",
  "status": "404"
}
```

B.5.4.2. ユーザーの削除が競合した

```
HTTP/1.1 409 Conflict
Content-Type: application/scim+json

{
  "schemas": ["urn:ietf:params:scim:api:messages:2.0:Error"],
  "detail": "SCIM サーバーがエラーの詳細を記入する",
  "status": "409"
}
```

B.5.4.3. ユーザー更新処理中にエラーが発生した

```
HTTP/1.1 500 Internal Server Error
Content-Type: application/scim+json

{
  "schemas": ["urn:ietf:params:scim:api:messages:2.0:Error"],
  "detail": "SCIM サーバーがエラーの詳細を記入する",
  "status": "500"
}
```

B.5.4.4. 認証に失敗した

```
HTTP/1.1 401 Unauthorized
Content-Type: application/scim+json

{
  "schemas": ["urn:ietf:params:scim:api:messages:2.0:Error"],
  "detail": "SCIM サーバーがエラーの詳細を記入する",
  "status": "401"
}
```

B.5.4.5. リソースへのアクセス・操作権限がない

```
HTTP/1.1 403 Forbidden
Content-Type: application/scim+json

{
  "schemas": ["urn:ietf:params:scim:api:messages:2.0:Error"],
  "detail": "SCIM サーバーがエラーの詳細を記入する",
  "status": "403"
}
```


付録C. サンプル実装例

C.1. TrustBind/Federation Manager を使った OP の実装

本章では、TrustBind/Federation Manager¹⁾を利用した OP の実装について述べる。本章で紹介するのは、バージョン 1.6.2 IDP Edition (OAS Limited Package) の OpenID Connect 機能を用いたものである。

なお、当実装ガイドで紹介した実装により相互接続性検証を実施し、以降で紹介する RP との相互接続を一通り確認している。

C.1.1. 実装概要

TrustBind/Federation Manager (以下、TrustBind/FM とする) は、様々な認証連携プロトコルを容易に導入できる認証連携モジュールである。

TrustBind/FM は、WebAP サーバー上で動作する Java アプリケーションであり、既存のクラウド利用企業のシステムと柔軟に連携して動作し、ID 連携を実現させることができるように設計されている。(図 C-1) そのため、ID 管理機能は既存のシステムに依存し、既存の認証基盤システム、あるいはデータストアと連携するための実装が必要となる。(図 C-2)

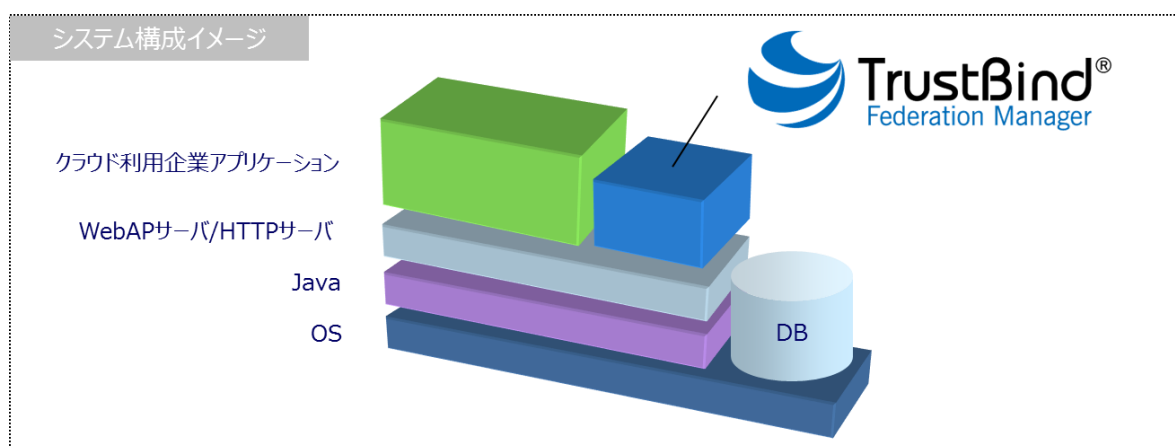


図 C-1: TrustBind/Federation Manager 利用時のシステム構成

¹⁾ TrustBind は NTT ソフトウェア株式会社の登録商標です。

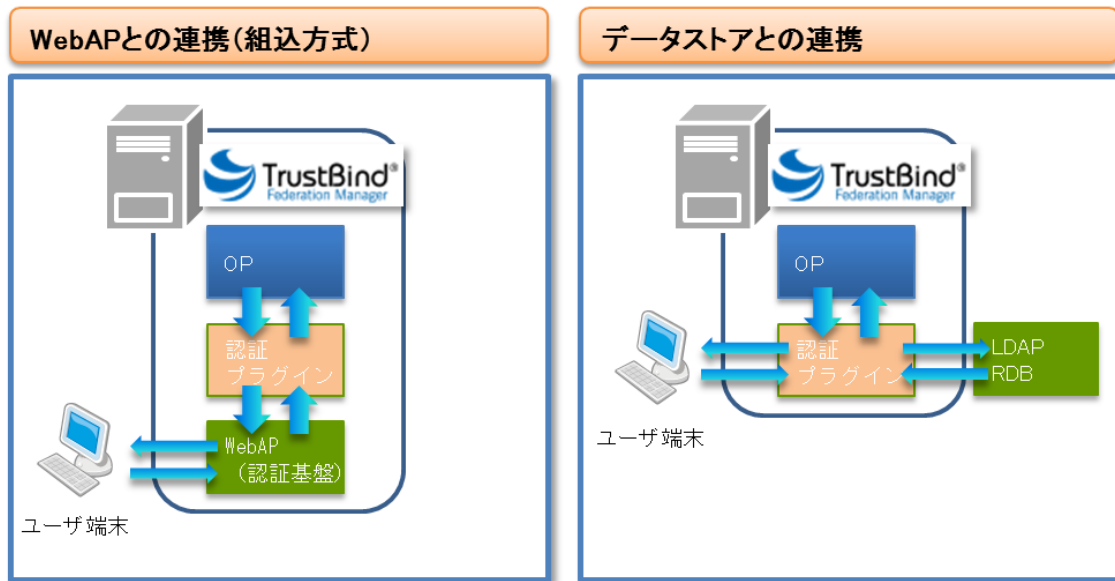


図 C-2: 既存システムとの連携方式

TrustBind/FMでは、認証処理と OpenID Connect プロトコル処理をそれぞれ別のコンポーネントに分離した構成としており、認証処理部分を認証プラグインと呼称している。認証プラグインは、プラグイン形式で様々なモジュールをアタッチメントすることが可能となっている。また、プラグイン内部では様々なカスタマイズが可能であり、クラウド利用企業の既存の認証をそのまま指定することも可能である。(図 C-2)

認可処理も認証処理と同様にプラグインを持ち、認可プラグインと呼称している。

これらを実装することで、当実装ガイドに沿った OP を実現できる。

認証プラグインは、TrustBind/FM の設定ファイル (oauth20.properties) の設定項目 APL_AUTH_PLUGIN に任意の URL に設定することで、TrustBind/FM からサーブレットの forward メソッドで呼び出され、独自の認証処理を行なうことが可能となる。

認可プラグインについても、認証プラグイン同様、TrustBind/FM の設定ファイル (oauth20.properties) の設定項目 APL_CONFIRM_PLUGIN に任意の URL に設定することで、TrustBind/FM からサーブレットの forward メソッドで呼び出され、独自の認可処理を行なうことが可能となる。

C.1.2. OP の実装

OP の実装として、クラウド利用企業のデータストアと連携する際の認証プラグイン、認可プラグインの実装概要を示す。

なお、当実装ガイドでは、入出力のパラメータなどの詳細な内容については記述しない。

C.1.2.1. 認証プラグインの実装

実装の概要図を以下に示す。

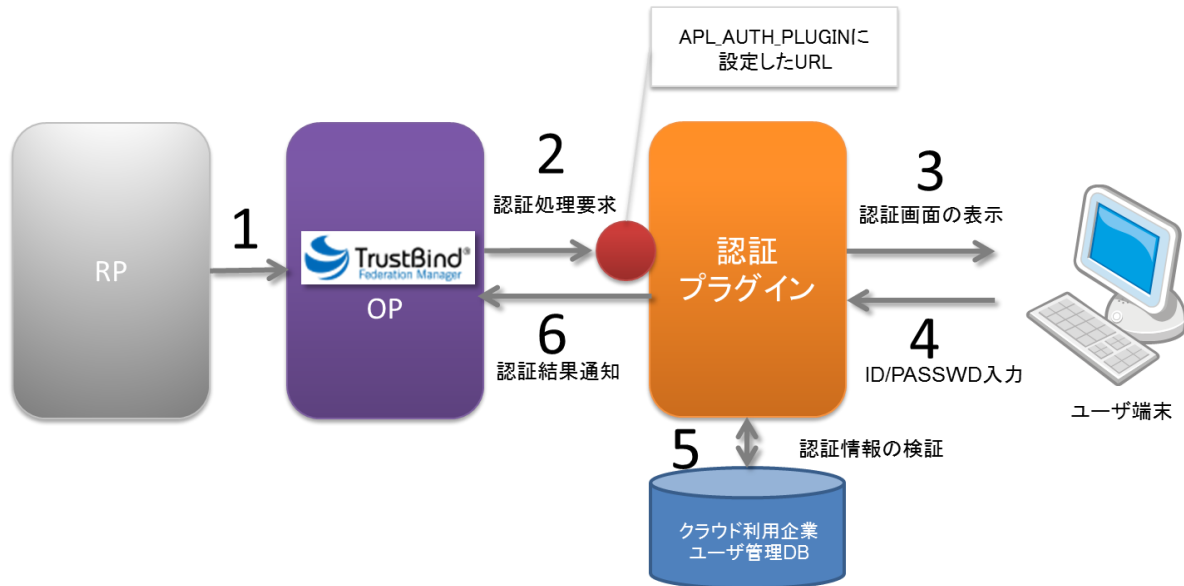


図 C-3: 認証プラグイン実装概要

認証プラグインで実現する機能は以下のとおりである。

- TrustBind/FM からのリクエストを受け取り、ユーザーに対して認証画面の表示を行なう
- 認証画面から入力された認証情報と、データストア登録されているクラウド利用企業のユーザー登録情報を比較し、認証結果を判定する
- 認証結果を TrustBind/FM に返却する

認証プラグインの実装は以上となる。

C.1.2.2. 認可プラグインの実装

実装の概要図を以下に示す。

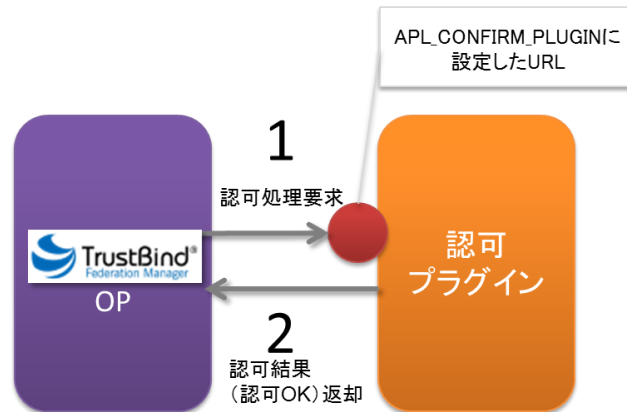


図 C-4: 認可プラグイン実装概要

認可プラグインで実現する機能は以下のとおりである。

- TrustBind/FM からの認可リクエストを受け取り、許可のレスポンスを返却する²

認可プラグインの実装は以上となる。

このように TrustBind/FM に用意されているプラグインに沿って実装を行なうことにより、非常に簡易に TrustBind/FM を当実装ガイドに沿った OP として動作させることが可能である。

² [5.1.4 節] ユーザーに認証連携の同意を得る を参照

C.2. OpenAM を使った OP の実装

本章では OpenAM を利用した OP の実装方法について述べる。

OpenAM は ForgeRock^[3]社が開発を行なっているアクセス管理を行なうオープンソースソフトウェアである。シングルサインオン機能やアクセス制御機能を主に提供し、OAuth 2.0 や OpenID Connect などの各種プロトコルにも対応しているのが特徴である。

C.2.1. OpenAM の導入

OpenAM の導入は、OpenAM の普及促進や情報交換を目的とした団体である OpenAM コンソーシアム^[4]が公開している技術 Tips 「OpenAM インストール手順」^[5]（以降、「情報ドキュメント」と呼ぶ）に従う。

C.2.1.1. 事前準備

まず情報ドキュメントの「4. 事前準備」までを実行する。以降、サーバーのホスト名、IP アドレスは情報ドキュメントと同じ値であるものとして説明する。

OpenAM は開発元の ForgeRock 社の Web サイトより入手できる^[6]。Nightly Build 版は最新のビルドのみがダウンロードできる。今回利用したのは以下の日付にビルドされたものとなる。

- 2015 年 8 月 4 日 (OpenAM-13.0.0-SNAPSHOT_20150804.war)

Nightly Build 版は随時更新されているため、バージョンによって設定コンソールの画面が変更されている場合がある。その場合は、当ドキュメントを参考に随時そのバージョンの操作画面に合わせて設定をして頂きたい。

C.2.1.2. Tomcat での TLS の有効化

情報ドキュメントの「5. Apache Tomcat セットアップ」に従ってアプリケーションコンテナである Tomcat のセットアップを行なう。

OpenID Connect の実装では TLS のサポートが必須ある。Tomcat での TLS の設定は公式ドキュメント^[7]などを参考に行なう。

³ ForgeRock <http://www.forgerock.com/>

⁴ OpenAM コンソーシアム <http://www.openam.jp/>

⁵ OpenAM コンソーシアム 技術 Tips Vol.1 「OpenAM インストール手順」
<http://www.openam.jp/category/member/techtips>

⁶ Downloads - ForgeRock Community <https://forgerock.org/downloads/>

⁷ Apache Tomcat 6.0 SSL Configuration HOW-TO <https://tomcat.apache.org/tomcat-6.0-doc/ssl-howto.html>

C.2.1.3. OpenAM のセットアップ

情報ドキュメントの「6. OpenAM セットアップ」に従って OpenAM のセットアップを行なう。

ただし、今回は TLS を有効にして設定を行なうため、ブラウザからアクセスする URL は `https://sso1.example.com:8443/openam/` とする。

また、ここに示す手順では、TLS で用いる証明書が自己署名証明書となるため、ブラウザによってはアクセス時にアクセスがブロックされることがある。その際は、今回の自己署名証明を受け入れるように設定が必要である。

C.2.2. OpenID Provider の設定

本節では、OpenAM を OP (OpenID Provider) として動作させ、RP (Relaying Party) を登録する設定の概要を述べる。

OpenAM では OAuth 2.0 の管理画面に従えば、OP の設定はほぼ自動で完了する。だが RP (Relaying Party) を登録する際はスコープ、リダイレクト URI、レスポンスタイプを設定する必要がある。

C.2.2.1. OP のセットアップ

OpenID Connect は OAuth 2.0 上で動くため、まず OAuth 2.0 プロバイダーとして設定する必要がある。

ブラウザ経由で OpenAM へ管理者ユーザーでログインすると、管理コンソール上に Realms という画面が表示される (図 C-5)。レルムは設定と管理対象のユーザーをまとめて管理するための OpenAM 上の仕組みで、初期時は Top Level Realm が用意されている。

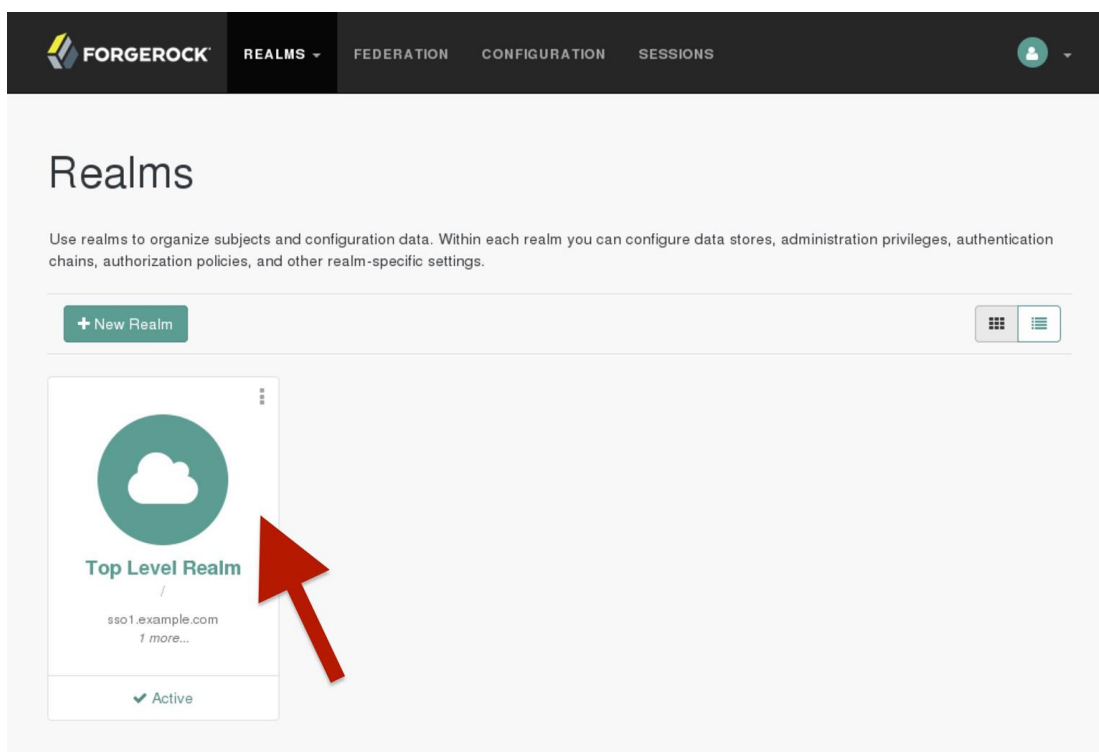


図 C-5: 管理者でログインした直後の管理コンソール

Top Level Realm のアイコンを選択すると、Realm Overview と呼ばれるレルム操作画面が表示される (図 C-6)。この「Configure OAuth2/OpenID Connect」のアイコンを選択し「OAuth2.0 の設定」画面を表示させる (図 C-7)。

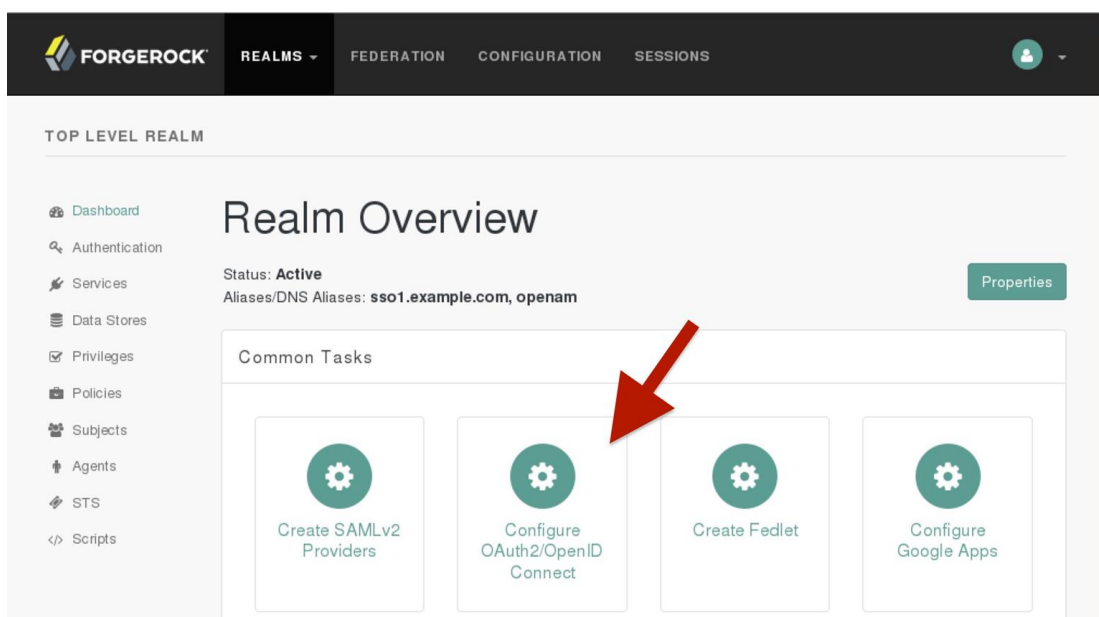


図 C-6: Realm Overview 画面



図 C-7: OAuth2 の設定画面

ここでは、デフォルト値のまま右上の [作成] を押してよい。ここで設定が成功すれば、成功を伝えるメッセージが表示される (図 C-8)。



図 C-8: 設定成功メッセージ

設定に成功すれば、OAuth 2.0 の認可サーバーとしてだけでなく OpenID Connect の OP としても動作しており <https://sso1.example.com:8443/openam/.well-known/openid-configuration> へアクセスすることで、以下のように OP の設定情報が JSON 形式で得られる。

```
{
  "response_types_supported": [
    "token id_token",
    "code token",
    "code token id_token",
    "token",
    "code id_token",
    "code",
    "id_token"
  ],
  "registration_endpoint": "https://sso1.example.com:8443/openam/oauth2/connect/register",
  "token_endpoint": "https://sso1.example.com:8443/openam/oauth2/access_token",
  "end_session_endpoint": "https://sso1.example.com:8443/openam/oauth2/connect/endSession",
  "version": "3.0",
  "userinfo_endpoint": "https://sso1.example.com:8443/openam/oauth2/userinfo",
  "subject_types_supported": [
    "public"
  ],
  "issuer": "https://sso1.example.com:8443/openam",
  "jwks_uri": "https://sso1.example.com:8443/openam/oauth2/connect/jwk_uri",
  "id_token_signing_alg_values_supported": [
    "HS256",
    "HS512",
    "RS256",
    "HS384"
  ],
  "check_session_iframe": "https://sso1.example.com:8443/openam/oauth2/connect/checkSession",
  "claims_supported": [
    "phone",
    "email",
    "address",
    "openid",
    "profile"
  ],
  "authorization_endpoint": "https://sso1.example.com:8443/openam/oauth2/authorize"
}
```


以上で OP としての設定は終了である。続いて、この OP を利用する RP を登録する必要がある。

また、OP の設定を変更したい場合は、再度 **Realm Overview** の画面まで戻って左側のメニュー内の **Services** を選択すれば、設定済みの OAuth 2.0 プロバイダーが表示され、そこから設定画面へ移動できる。

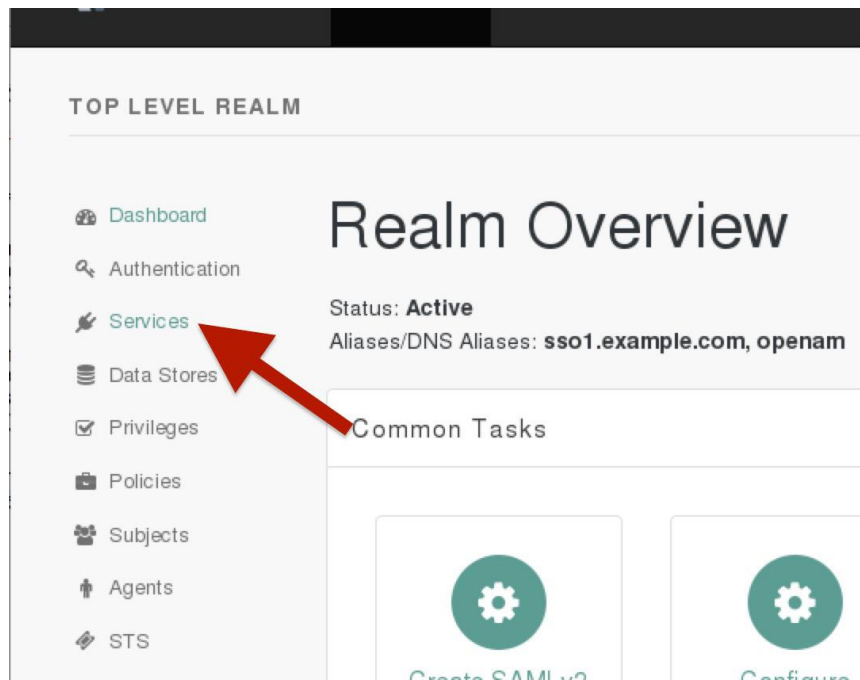


図 C-9: OAuth 2.0 プロバイダー一覧表示

C.2.2.2. Relaying Party の登録

次に RP (Relaying Party) の情報を OP として設定した OpenAM に登録する。

OP は RP からの認証リクエストに含まれるスコープ、レスポンスタイプ、クライアント ID、リダイレクト URI などを検証する。そのため、予め RP についてこれらの情報を登録しておく必要がある。

RP は OAuth 2.0 のクライアントとして登録する。管理者で OpenAM にログインし、先程 OP を作成したレルムを選択する (図 C-10)。ここでは Top Level Realm を選択している。

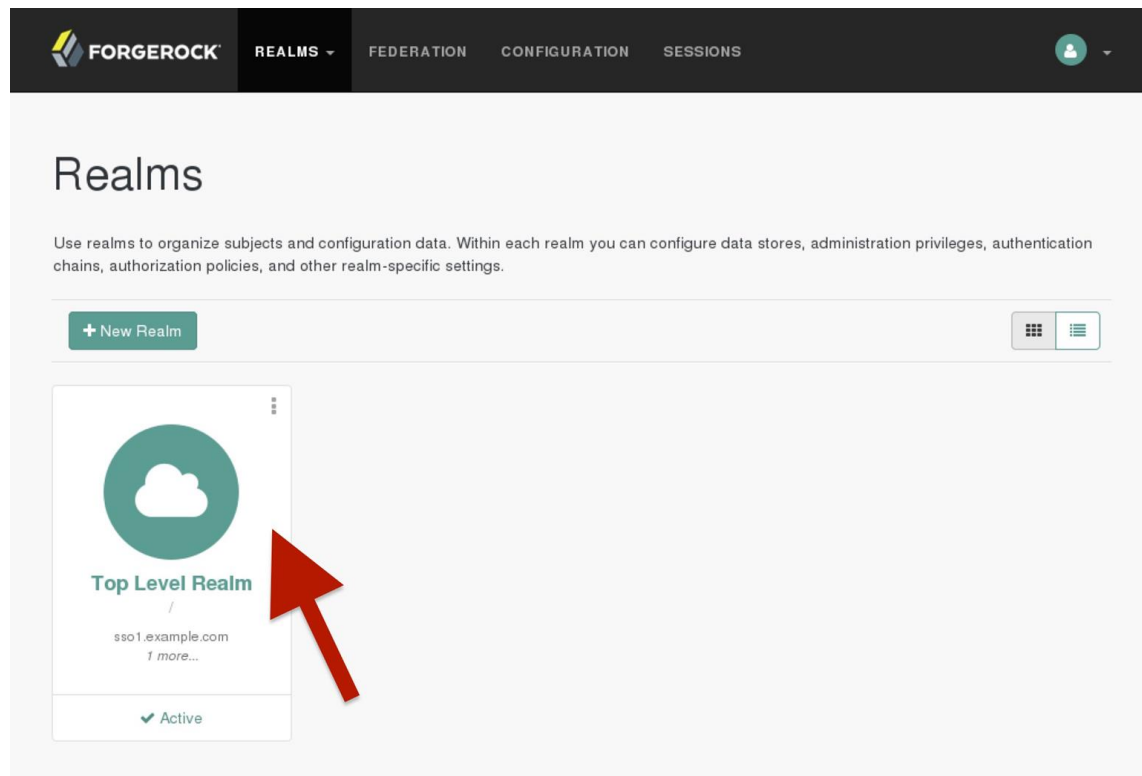


図 C-10: レalm選択

レalmを選択すると Realm Overview 画面が表示されるので、左メニューの Services を選択する (図 C-9)。

その後、[エージェント] タブ → [OAuth2.0 クライアント] タブの順に選択する。ここでエージェントのリストの [新規] ボタンを押して OAuth2.0 クライアントの作成画面に移動する (図 C-11)。



図 C-11: OAuth 2.0 クライアントの作成画面への移動手順

作成する際にはクライアントの ID とパスワードの入力が必要である。これらはトークンエンドポイントでクライアントを認証する場合に用いられる。入力が完了したら右上の [作成] ボタンを押す (図 C-12)。

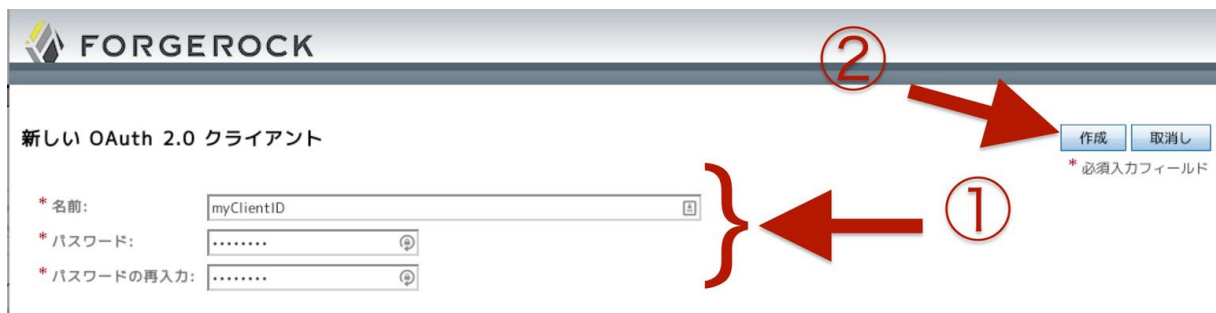


図 C-12: 新しい OAuth 2.0 クライアントの追加

作成が成功すると、エージェントのリストに今作成したエージェントが追加されているので、それを選択する (図 C-13)。



図 C-13: 設定を変更するクライアントの選択

選択後、クライアントの設定画面（図 C-14）で必要な設定を行なう。



図 C-14: クライアント設定画面

登録した OAuth 2.0 クライアントを OpenID Connect に対応させるためにいくつかの設定を行なう必要がある。最低限設定が必要な項目は、スコープ、リダイレクト URI、レスポンスタイプの 3 つである。

OpenID Connect の場合、リクエスト時のスコープに “openid” が必ず指定される。RP のデフォルト設定ではスコープが未設定となっている。そのため、最低限スコープに “openid” を追加する必要がある。その他に必要なスコープ (profile など) があれば適宜追加する。スコープを登録するには新しい値に登録したいスコープを入力し [追加] ボタンを押す (図 C-15)。追加を押すと、[現在の値] リストに入力値が追加される。これを登録する必要があるスコープの分だけ繰り返す。

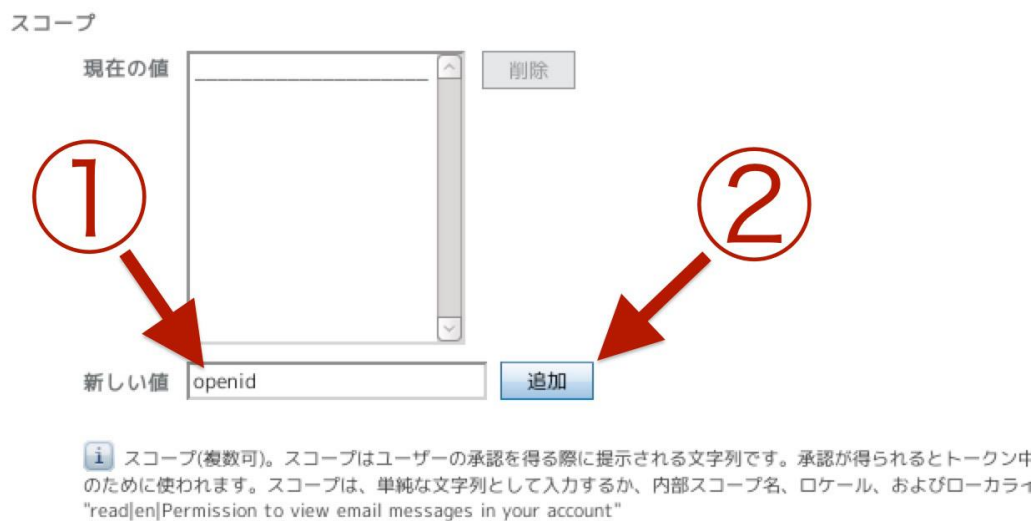


図 C-15: スコープの登録

また、リダイレクト URI も初期設定では未設定となっているので設定が必要である。

レスポンスタイプはデフォルトで “code” や “id_token” などが既に登録済みである。もし、不足がある場合は適宜追記する。

必要な設定項目の入力が終了したら設定画面右上の [保存] ボタンを押す。このボタンを押すまでは各入力値は設定に反映されない。

以上で、RP の登録が終了である。

OpenAM における OpenID Connect のより詳しい設定については、ForgeRock のドキュメントが参考になる⁸⁾。

⁸⁾ OpenAM Administration Guide, 14. Managing OpenID Connect 1.0 Authorization - <http://openam.forgerock.org/doc/bootstrap/admin-guide/index.html#chap-openid-connect>

C.3. Ruby による OP サーバーのスクラッチ実装例

C.3.1. 実装概要

サンプルとして作成した OP サーバーの概要は以下の通り。

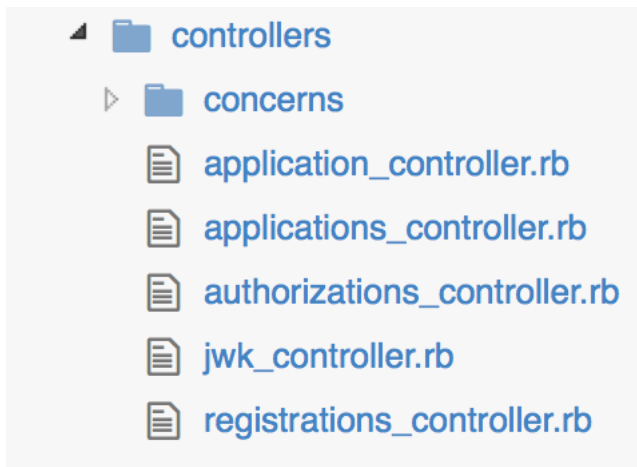
- 動作環境
 - Ruby 2.1.5
 - Ruby on Rails 4.1.5
 - SQLite3
 - CentOS 6.5
- 動作概要
 - Web フォームからユーザーの登録ができる
 - Implicit Flow で OpenID Connect を使った認証ができる
 - UserInfo エンドポイントは実装せず、ID トークンの claim として返却する
 - ID トークンは RS256 アルゴリズムで署名しているため、jwks エンドポイントを提供する
 - ユーザー認証部分の処理に関しては簡略化のため、devise という gem ライブラリを利用している

C.3.2. プログラムサンプル

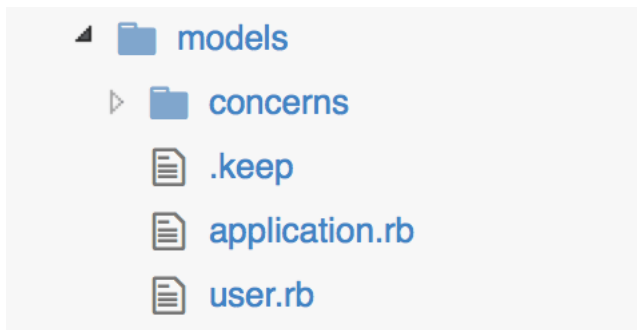
Rails アプリケーションとして以下の URL よりダウンロードできる。

- <https://github.com/openid-foundation-japan/eiwig-guideline-samples/sample-ruby-op>

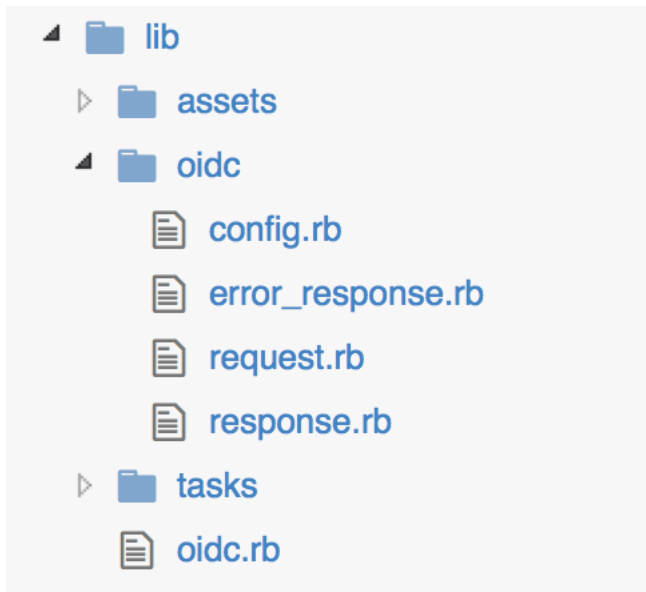
C.3.3. プログラム構成



| コントローラ名 | 概要 |
|---------------------------|------------------|
| applications_controller | RP 情報へのリクエストを処理 |
| authorizations_controller | AuthN/Z リクエストを処理 |
| jwk_controller | jwk 情報へのリクエストを処理 |
| registrations_controller | ユーザー情報へのリクエストを処理 |



| モデル名 | 概要 |
|-------------|------------|
| application | RP 情報のモデル |
| user | ユーザー情報のモデル |



| ライブラリ名 | 概要 |
|---------------------|------------------------------|
| oidc/config | OP に関する設定情報を定義 |
| oidc/error_response | AuthN/Z リクエストに対するエラーレスポンスを定義 |
| oidc/request | AuthN/Z リクエストに対する処理を定義 |
| oidc/response | AuthN/Z レスポンスに対する処理を定義 |

C.3.4. 動作確認方法

1. インストールとアプリケーションの起動

```
$ cd <APP_ROOT>
$ bundle exec rake db:migrate
$ bundle exec rails s
```

<http://localhost:3000/> にアクセスする。

2. ユーザーの登録

Sign in

Email

Password

Remember me

[Sign up](#)

アプリケーションの URL へアクセスすると `sign_in` 画面へリダイレクトされ、上記画面が表示される。`Sign up` のリンクをクリックし、まずユーザーを登録する。

3. RP の登録

Welcome! You have signed up successfully.

Listing applications

[Sign out](#)

Name

[New Application](#)

サインアップ後、上記ページが表示されるので、`New Application` のリンクをクリックして新規 RP を登録する。

New application

Name

Redirect uri

[Back](#)

RP を登録すると AuthN/Z リクエストに必要な情報が表示される。

Application was successfully created.

Name: test_rp1

Client: [REDACTED]

Client secret: [REDACTED]

Redirect uri: http://localhost:9000/callback

C.3.5. 簡単な OP の仕様説明

1. サポートしている AuthN/Z リクエストパラメータ

| パラメータ | 内容 |
|---------------|--|
| scope | [必須] openid を含むこと。openid, profile, email をサポート。 |
| response_type | [必須] id_token のみサポート。 |
| client_id | [必須] RP のクライアント識別子。 |
| redirect_uri | [必須] レスポンスが返されるリダイレクト先 URI 文字列。RP 登録時の URI 群のいずれかに完全一致する必要がある。 |
| state | [必須] CSRF 対策の文字列。 |
| nonce | [必須] リプレイ攻撃対策の文字列。指定した文字列が ID トークンの nonce フィールドに格納される。 |

2. レスポンスパラメータ

| パラメータ | 内容 |
|------------|---------------------------------------|
| token_type | Bearer 固定 |
| id_token | ID トークン文字列。 |
| state | 認可リクエストで state を指定していた場合のみ、その値が格納される。 |

3. ID トークンに含まれるクレーム

| クレーム名 | 内容 |
|-------|---------------------------|
| iss | ID トークンを発行した OP の識別子。文字列。 |
| sub | ユーザーの識別子。 |
| aud | この ID トークンの発行対象の識別子の配列。 |

| | |
|------------------|--|
| exp | ID トークンが実行する時刻。整数値。 |
| iat | ID トークンが発行された時刻。整数値。 |
| nonce | [nonce を指定したリクエストの場合のみ] リプレイ攻撃対策として使われる文字列。 |
| name | [scope に profile を指定したリクエストの場合のみ] End-User の表示用フルネーム。 |
| given_name | [scope に profile を指定したリクエストの場合のみ] End-User の名 (given name / first name)。 |
| family_name | [scope に profile を指定したリクエストの場合のみ] End-User の姓 (surname / last name)。 |
| given_name_kana | [scope に profile を指定したリクエストの場合のみ] End-User の名(かな)。 |
| family_name_kana | [scope に profile を指定したリクエストの場合のみ] End-User の姓(かな)。 |
| gender | [scope に profile を指定したリクエストの場合のみ] End-User の性別。 |
| email | [scope に email を指定したリクエストの場合のみ] End-User の選好する Email アドレス。 |

動作方法やソースコードの詳細については、ダウンロードしたプログラムソースおよび README を参照していただきたい。

C.4. Ruby による RP サーバーのスクラッチ実装例

C.4.1. 実装概要

サンプルとして作成した RP サーバーの概要は以下の通り。

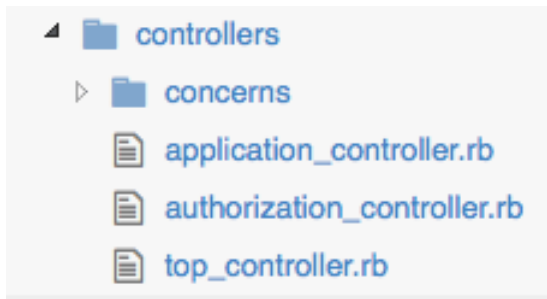
- 動作環境
 - Ruby 2.1.5
 - Ruby on Rails 4.1.5
 - SQLite3
 - CentOS 6.5
- 動作概要
 - Implicit Flow で TrustBind, OpenAM, スクラッチ OP にログインできる
 - 同じネットワーク内に存在していると仮定する SCIM サーバーに対してユーザー識別子を key に問い合わせし、ユーザー情報を取得できる
 - 実装の簡略化のため https://github.com/nov/openid_connect の gem ライブラリを利用している

C.4.2. プログラムサンプル

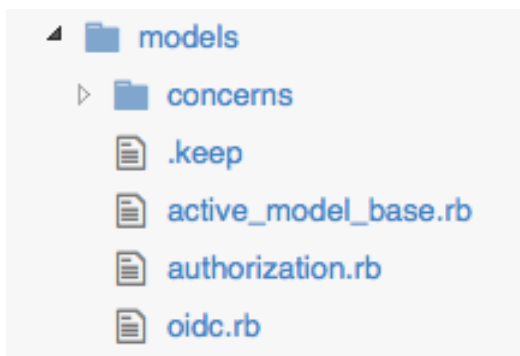
Rails アプリケーションとして以下の URL よりダウンロードできる。

- <https://github.com/openid-foundation-japan/eiwg-guideline-samples/sample-ruby-rp>

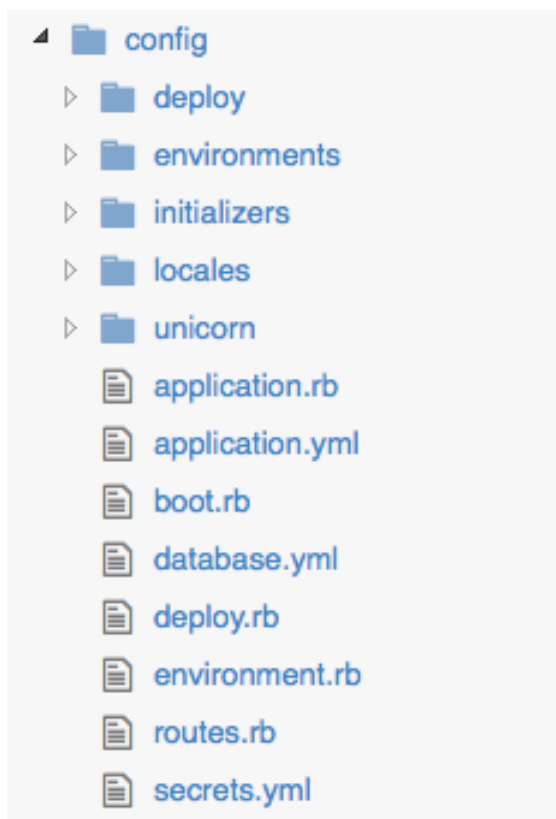
C.4.3. プログラム構成



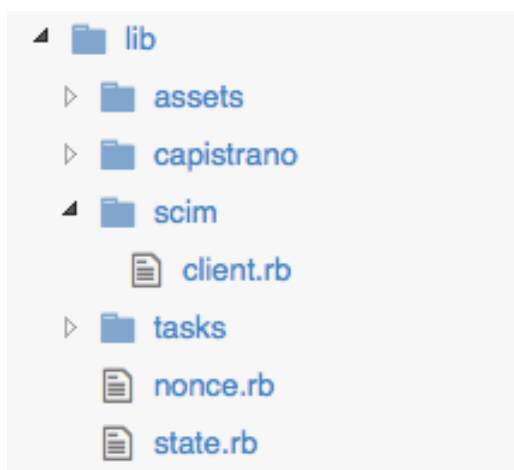
| コントローラファイル名 | 概要 |
|-----------------------------|------------------------|
| authorization_controller.rb | OP への AuthN/Z リクエストを処理 |
| top_controller.rb | ログイン前後の画面へのリクエストを処理 |



| モデルファイル名 | 概要 |
|---------------------|----------------------------|
| active_mode_base.rb | 共通処理を定義 |
| authorization.rb | AuthN/AuthZ リクエストをモデル化したもの |
| oidc.rb | AuthN/Z レスポンスをモデル化したもの |



| 設定ファイル名 | 概要 |
|-----------------|----------------------|
| application.yml | OP にリクエストするための設定値を定義 |



| クラスファイル名 | 概要 |
|----------------|-----------------------|
| scim/client.rb | SCIM サーバーに対するリクエストを定義 |

C.4.4. 動作確認方法

1. インストール

```
$ cd <APP_ROOT>
$ bundle install --path=vendor/bundle
$ bundle exec rake db:migrate
```

2. 設定ファイルの編集 / UserInfo が不要な場合 (ID トークンの claim として含まれている等)

```
production:
  <<: *defaults
  scim:
    server: <SCIM サーバーの URL (e.g. http://192.168.0.1:8080/)>
    auth:
      user: <SCIM サーバーの認証情報/アカウント>
      password: <SCIM サーバーの認証情報/アカウント>
  oidc:
    corpA:
      issuer: <OP の issuer 値>
      identifier: <OP から発行された client_id>
      authorization_endpoint: <OP が提供する Authorization エンドポイント>
      jwks_uri: <OP が提供する jwks_uri エンドポイント>
      redirect_uri: <OP に事前登録済みのリダイレクト URI>
    corpB
    ...
    corpC
    ...
```

3. 設定ファイルの編集 / UserInfo エンドポイントが必要な場合

```
production:
  <<: *defaults
    scim:
      server: <SCIM サーバーの URL (e.g. http://192.168.0.1:8080/)>
      auth:
        user: <SCIM サーバーの認証情報/アカウント>
        password: <SCIM サーバーの認証情報/アカウント>
    oidc:
      corpA:
        ...
      corpB
        issuer: <OP の issuer 値>
        identifier: <OP から発行された client_id>
        authorization_endpoint: <OP が提供する Authorization エンドポイント>
        jwks_uri : <OP が提供する jwks_uri エンドポイント>
        userinfo_endpoint: <OP が提供する userinfo エンドポイント>
        redirect_uri: <OP に事前登録済みのリダイレクト URI>
      corpC
        ...
```

4. 設定ファイルの編集 / ID トークンの検証に必要な公開鍵情報をローカルで保持する場合

```
production:
  <<: *defaults
    scim:
      server: <SCIM サーバーの URL (e.g. http://192.168.0.1:8080/)>
      auth:
        user: <SCIM サーバーの認証情報/アカウント>
        password: <SCIM サーバーの認証情報/アカウント>
    oidc:
      corpA:
        ...
      corpB
        ...
      corpC
        issuer: <OP の issuer 値>
        identifier: <OP から発行された client_id>
        authorization_endpoint: <OP が提供する Authorization エンドポイント>
        userinfo_endpoint: <OP が提供している userinfo エンドポイント>
        redirect_uri: <OP に事前登録済みのリダイレクト URI>
        x509_signing_key: <X.509 形式の証明書の文字列>
```


5. アプリケーションの起動

```
$ bundle exec rails s
```

<http://localhost:3000/> にアクセスする。

6. TOP ページの表示

OpenID Connect

[corpA\(RubyOP\) でログイン](#) / [corpB\(OpenAM\) でログイン](#) / [corpC\(TrustBind\) でログイン](#)

TOP 画面にアクセスするとログインする Provider (OP) が選択できる。

それぞれリンクをクリックすると設定された内容にしたがって、Authorization リクエストが発行される。

You need to sign in or sign up before continuing.

Sign in

Email

Password

Remember me

動作方法やソースコードの詳細については、ダウンロードしたプログラムソースおよび README を参照していただきたい。

C.5. mod_auth_openidc を使った RP の実装

mod_auth_openidc は、Ping Identity 社がオープンソースソフトウェアとして GitHub 上に公開している、Apache 2.x HTTP Server を OpenID Connect の RP として動作させる事を可能にする認証モジュールである。

本モジュールでは指定したパスにあるコンテンツを OpenID Connect の認証により保護し、OP から発行された ID Token に含まれる認証情報等を HTTP にヘッダに乗せ、後続の Web アプリケーションに連携することが可能である。

そのため既存の Web アプリケーションの前段にて設定を行なうことで、アプリケーションへの改変も少なく RP とすることも可能である。

C.5.1. インストール

GitHub から mod_auth_openidc をダウンロードし、内包されている INSTALL ファイルを参照し、インストールを行なう。

- https://github.com/pingidentity/mod_auth_openidc

C.5.2. 設定 (httpd.conf)

mod_auth_openidc の設定は、httpd.conf またはそこから参照されるファイルに記載する必要があり、設定の詳細は GitHub 上の mod_auth_openidc に存在する auth_openidc_conf を参考に行なう。

- https://github.com/pingidentity/mod_auth_openidc/blob/master/auth_openidc.conf

RP として動作させたい範囲に対して下記設定を行なう。

```
<Location />
  AuthType openid-connect #openid connect の認証が必要であることの指定
  Require valid-user #認証済みユーザーであればアクセス可能
</Location>
```

本実装ガイドの通りの実装を行なうにあたって必要な設定は以下の通りである。

1. MetadataURL の指定

OP 側で Metadata URL を提供している場合、その URL を指定する。

```
OIDCProviderMetadataURL https://op.com.example.co.jp/.well-known/openid-configuration
```

2. Issuer の指定

Issuer の識別子を指定する。

```
OIDCProviderIssuer https://op.com.example.co.jp
```

3. AuthorizationEndpoint の指定

OP の認証エンドポイント URL を指定する。

```
OIDCProviderAuthorizationEndpoint https://op.com.example.co.jp/authorize
```

4. client_id の指定

OP に設定したクラウドサービスに割り当てた client_id を指定する。

```
OIDCClientID pWBoRam9sG
```

5. client_secret の指定

本実装ガイドで利用しているフローは **Implicit Flow** であるため、client_secret は不要であるが、当モジュール内では設定が必須となっているため、値を設定する。

```
OIDCClientSecret secret
```

6. 署名アルゴリズムの指定

ID Token の署名に利用する署名アルゴリズムを指定する。

```
OIDCIDTokenSignedResponseAlg RS256
```

7. JWKS URI の指定

OP 側で JWKS URI を提供している場合、その URI を指定する。

```
OIDCProviderJwksUri https://op.com.example.co.jp/jwks
```

8. PEM の指定

OP 側から署名検証用の公開鍵が PEM 形式のファイルとして渡されている場合、PEM そのファイル名を指定する。以下の例は、`/etc/httpd/conf/myCert.pem` という名前でファイルを保管している場合の設定である。

```
OIDCPublicKeyFiles /etc/httpd/conf/myCert.pem
```

9. RedirectURI の指定

OP 認証後に ID Token を送付するための Redirect URI を指定する。`mod_auth_openidc` の設定で RP としての対象となっている URI を指定する必要がある。

```
OIDCRedirectURI https://www.svc.example.net/cb
```

10. ResponseType の設定

ResponseType を指定。本実装ガイド内では ImplicitFlow で ID Token の送付が必要なため `id_token` のみを指定する。

```
OIDCResponseType id_token
```

11. Scope の設定

Scope を指定。本実装ガイド内では ID Token を応答すれば十分のため、`openid` のみを指定する。

```
OIDCScope openid
```

12. 認証クッキーの暗号パスフレーズの設定

認証後に発行される `mod_auth_openidc_session` クッキーの暗号化パスフレーズを指定する。

```
OIDCCryptoPassphrase passphrase
```

C.5.3. 実装上の注意

ガイド内に記載している再認証の処理であるが、再認証については RP から下記の値を付加した上で OP に向けて認証要求を送信する必要がある。

- prompt=login
- maxage=30
- login_hint=(プロビジョニング時に externalUserName 属性で渡された値)

現時点において、`mod_auth_openidc` では、認証後に再認証を求めるためのエンドポイント及び、状況に応じたパラメータの変更が難しいため、再認証に関する処理を行なう場合はアプリケーション側で実装を行なう必要がある。

C.5.4. サンプル設定

```
LoadModule auth_openidc_module modules/mod_auth_openidc.so
# OIDCProviderMetadataURL https://op.com.example.co.jp/.well-known/openid-configuration

OIDCProviderIssuer https://op.com.example.co.jp
OIDCProviderAuthorizationEndpoint https://op.com.example.co.jp/authorize
OIDCProviderJwksUri https://op.com.example.co.jp/jwks
OIDCPublicKeyFiles /etc/httpd/conf/myCert.pem

OIDCClientID pWBoRam9sG
OIDCClientSecret secret
OIDCIDTokenSignedResponseAlg RS256

OIDCRedirectURI https://www.svc.example.net/cb
OIDCResponseType id_token
OIDCScope openid

OIDCSSLValidateServer on
OIDCCryptoPassphrase passphrase

<Location /example/>
  AuthType openid-connect
  Require valid-user
</Location>
```

C.6. Java による SCIM サーバーのスクラッチ実装例

C.6.1. 実装概要

サンプルとして作成した SCIM サーバーの概要は以下の通り。

- 開発環境は eclipse による Tomcat プロジェクトの Java サブレット
- 動作環境は Tomcat7 + Java7
- JSON の解析には JACKSON を利用
- 1 アプリケーション 1 テナント
- リソース情報はローカルファイルから読み込みオンメモリで保持
- ユーザーリソースの検索 (GET) / 追加 (POST) / 更新 (PUT) / 削除 (DELETE) を実装

C.6.2. プログラムサンプル

eclipse プロジェクトとして以下の URL よりダウンロードできる。

- <https://github.com/openid-foundation-japan/eiwg-guideline-samples/sample-scim-server>

C.6.3. 利用方法

- eclipse へのプロジェクトインポート方法
 1. eclipse のメニューから [ファイル]→[インポート] を選択する。
 2. 選択画面から [一般]→[既存プロジェクトをワークスペースへ] を選択する。
 3. インポート画面の [ルートディレクトリの選択] テキストボックスに、sample-scim-server フォルダを入力する。
 4. インポート画面の [プロジェクト] に表示される scim プロジェクトを選択する。
 5. [完了] ボタンを押下してプロジェクトをインポートする。


















利用する eclipse のバージョンによっては画面の表示名称が異なる場合がある。使用している eclipse に合わせて適宜読み替えていただきたい。

- SCIM サーバーの利用法
 1. `eiwg.war` ファイルを Tomcat に配備する。(war ファイルの配備方法は Tomcat のドキュメント等を参照)
 2. `WEB-INF` フォルダ直下に、各種設定ファイルが配置される。管理者の情報や、スキーマ情報、初期ユーザー情報等を変更する場合は、設定ファイル変更後に Web アプリケーションを再起動する。
 3. BaseURL は 「`http://<ご利用のサーバー名>:<ご利用のポート番号>/eiwg/scim`」 である。curl コマンド等の HTTP クライアントを利用して各種エンドポイントへアクセスして利用する。

curl コマンドによるユーザーリソース取得の例

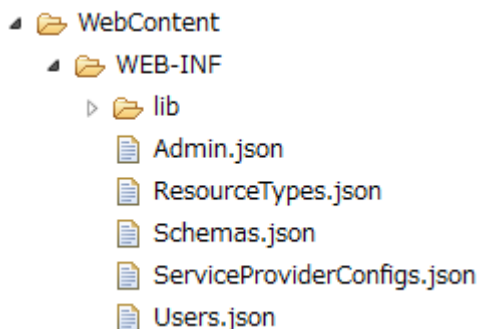
```
$ curl http://scim.svc.example.net:8080/eiwg/scim/Users -H "Authorization: Bearer MDNhNT1hN2ItYmE1NC00OWQ3LWFkMzQtODliYjhhN2Q0Mj1h"
```

C.6.4. プログラム構成

- モジュール構成
 - ▾  src
 - ▾  jp.or.openid.eiwg.constants
 - ▷  MessageConstants.java
 - ▾  jp.or.openid.eiwg.filter
 - ▷  InitFilter.java
 - ▾  jp.or.openid.eiwg.listener
 - ▷  InitListener.java
 - ▾  jp.or.openid.eiwg.scim.operation
 - ▷  Operation.java
 - ▾  jp.or.openid.eiwg.scim.servlet
 - ▷  ResourceTypes.java
 - ▷  Schemas.java
 - ▷  ServiceProviderConfigs.java
 - ▷  Users.java
 - ▾  jp.or.openid.eiwg.scim.util
 - ▷  SCIMUtil.java
 - ▷  SCIMUtilException.java

| クラス名 | 概要 |
|------------------------|--|
| MessageConstants | エラーメッセージを定義したクラス |
| InitFilter | サーブレット実行前にパスのチェックを行なうフィルタクラス |
| InitListener | アプリケーション起動時に各種設定ファイルの読み込みを行なうリスナークラス |
| Operation | 認証や各種リソースに対する操作を実装したクラス |
| ResourceTypes | エンドポイント <code>ResourceTypes</code> のサーブレットクラス |
| Schemas | エンドポイント <code>Schemas</code> のサーブレットクラス |
| ServiceProviderConfigs | エンドポイント <code>ServiceProviderConfigs</code> のサーブレットクラス |
| Users | エンドポイント <code>ServiceProviderConfigs</code> のサーブレットクラス |
| SCIMUtil | フィルター解析,リソース情報検索など部品的な処理を実装したクラス |
| SCIMUtilException | 上記 <code>SCIMUtil</code> の例外クラス |

- 設定ファイル



| ファイル名 | 説明 |
|-----------------------------|--|
| Admin.json | 管理者情報(scim クライアントの認証情報) |
| ResourceTypes.json | エンドポイント ResourceTypes のリソース情報 |
| Schemas.json | エンドポイント Schemas のリソース情報 |
| ServiceProviderConfigs.json | エンドポイント ServiceProviderConfigs のリソース情報 |
| Users.json | エンドポイント Users のリソース情報 |

C.6.5. ポイント

1. InitListener クラス

`ServletContextListener` の実装クラス。Web アプリケーション起動時の処理として、各設定ファイルの読み込みを行ない、読み込んだ情報を各サーブレットで参照するためにコンテキスト属性として保持する。

各設定ファイルの JSON データの読み込みには、Jackson の `ObjectMapper` を利用している。

2. InitFilter クラス

`Filter` の実装クラス。各サーブレットの実行前に、サポートしているエンドポイントに対するアクセスかどうかのチェックを行なっている。

3. サーブレットクラス

各エンドポイントに対する操作を実際に処理する `HttpServlet` を継承したクラス。

サポートする各メソッド (`doGet()`、`doPost()`、`doPut()`、`doDelete()`) をオーバーライドして実際の処理を記述している。

PATCH メソッドを実装する場合は、`HttpServlet` の標準でサポートされていないため、自前でメソッドを判定して処理を振り分ける（`service` メソッドをオーバーライドして実装する）必要がある点に注意が必要である。

各メソッドは、大きく分けて以下の 3 つの処理を行なっている。

- 認証処理
- リクエストに対する処理
- レスポンスの生成処理

認証処理は `Operation` クラスの `Authentication()` メソッドとして実装している。

また、各リクエストに対する処理は、

- 追加処理は `Operation` クラスの `createUserInfo()` メソッド
- 検索処理は `Operation` クラスの `searchUserInfo()` メソッド
- 更新処理は `Operation` クラスの `updateUserInfo()` メソッド
- 削除処理は `Operation` クラスの `deleteUserInfo()` メソッド

として実装している。

詳しくは、ダウンロードしたプログラムソースを参照していただきたい。

著者一覧

| | |
|--------|---------------------------------------|
| 飯野 亨太 | オープンソース・ソリューション・テクノロジー株式会社 |
| 上田 尊教 | エクスジェン・ネットワークス株式会社 |
| 宇佐美 友也 | 株式会社インターネットイニシアティブ |
| 氏繩 武尊 | 株式会社オーグス総研 |
| 桑田 雅彦 | 日本電気株式会社 |
| 坂本 一仁 | セコム株式会社 |
| 作田 宗臣 | エヌ・ティ・ティ・ソフトウェア株式会社 |
| 真武 信和 | 一般社団法人 OpenID ファウンデーション・ジャパン エバンジェリスト |
| 八幡 孝 | 株式会社オーグス総研 |

EIWG 技術タスクフォース参加者一覧

| | |
|--------|---------------------------------------|
| 飯野 亨太 | オープンソース・ソリューション・テクノロジー株式会社 |
| 上田 尊教 | エクスジェン・ネットワークス株式会社 |
| 宇佐美 友也 | 株式会社インターネットイニシアティブ |
| 氏繩 武尊 | 株式会社オーグス総研 |
| 桑田 雅彦 | 日本電気株式会社 |
| 坂本 一仁 | セコム株式会社 |
| 作田 宗臣 | エヌ・ティ・ティ・ソフトウェア株式会社 |
| 中尾 和弘 | 株式会社オーグス総研 |
| 野村 健太郎 | エクスジェン・ネットワークス株式会社 |
| 浜口 優 | 株式会社オーグス総研 |
| 真武 信和 | 一般社団法人 OpenID ファウンデーション・ジャパン エバンジェリスト |
| 八幡 孝 | 株式会社オーグス総研 |