Red Hat Single Sign-On 7.6

アップグレードガイド

Red Hat Single Sign-On 7.6 向け

Last Updated: 2022-08-07

Red Hat Single Sign-On 7.6 アップグレードガイド

Red Hat Single Sign-On 7.6 向け

Enter your first name here. Enter your surname here.
Enter your organisation's name here. Enter your organisational division here.
Enter your email address here.

法律上の通知

概要

本ガイドは、以前のバージョンの Red Hat Single Sign-On 7.6 からアプリケーションのアップグレードを説明します。

目次

多様性を受け入れるオープンソースの強化

Red Hat では、コード、ドキュメント、Web プロパティーにおける配慮に欠ける用語の置き換えに取り組んでいます。まずは、マスター (master)、スレーブ (slave)、ブラックリスト (blacklist)、ホワイトリスト (whitelist) の 4 つの用語の置き換えから始めます。これは大規模な取り組みであるため、これらの変更は今後の複数のリリースで段階的に実施されます。詳細は、Red Hat CTO である Chris Wright のメッセージ をご覧ください。

第1章 RED HAT SINGLE SIGN-ON のアップグレード

Red Hat Single Sign-On (RH-SSO) 7.6 は Keycloak プロジェクトをベースとしており、SAML 2.0、OpenID Connect、OAuth 2.0 などの一般的な標準をベースとした Web シングルサインオン機能を提供することで、Web アプリケーションのセキュリティーを提供します。Red Hat Single Sign-On Server は SAML または OpenID Connect ベースのアイデンティティープロバイダーとして機能することが可能で、標準ベースのトークンを使用して、エンタープライズユーザーディレクトリーまたはサードパーティー SSO プロバイダーとアプリケーションを仲介します。

RH-SSO は、スタンドアロンサーバーまたは管理対象ドメインである 2 つの操作モードを提供します。スタンドアロンサーバーの操作モードは、単一のサーバーインスタンスとして RH-SSO の実行を表します。管理対象ドメインの操作モードを使用すると、単一の制御ポイントから複数の RH-SSO インスタンスを管理できます。アップグレードプロセスは、実装された操作モードによって異なります。該当する場合は、各モードの具体的な手順が示されています。

本ガイドの目的は、Red Hat Single Sign-On 7.x から Red Hat Single Sign-On 7.6 へのアップグレードに必要な手順を文書化することです。

1.1. アップグレードについて

RH-SSO のバージョンに応じて、3 種類のアップグレードを選択します。ただし、Keycloak から開始する場合は、この手順 を選択します。

1.1.1. メジャーアップグレード

Red Hat Single Sign-On 7.2 から Red Hat Single Sign-On 8.0 など、RH-SSO を別のメジャーリリースにアップグレードする場合は、メジャーアップグレードまたは移行が必要になります。メジャーリリース間で API の重大な変更があり、アプリケーションまたはサーバーの拡張機能の一部を書き換えが必要になる場合があります。

1.1.2. マイナーアップデート

Red Hat Single Sign-On は、定期的にポイントリリースを提供します。これは、バグ修正、セキュリティー修正、および新機能が含まれるマイナーアップデートです。Red Hat Single Sign-On ポイントリリースを別のリリースにアップグレードする場合 (たとえば、Red Hat Single Sign-On 7.3 から Red Hat Single Sign-On 7.6 へ) は、プライベートではないまたはサポート対象外のもの、またはテクノロジープレビュー API が使用されてる場合は、アプリケーションやカスタムサーバーの拡張機能にコードを変更する必要はありません。

1.1.3. マイクロアップデート

Red Hat Single Sign-On 7.6 では、バグ修正やセキュリティー修正が含まれるマイクロリリースを定期的に提供します。マイクロリリースは、7.6.0 から 7.6.1 など、最後の数字のマイナーリリースバージョンを増分します。これらのリリースには移行は必要なく、サーバー設定ファイルには影響を及ぼしません。ZIP インストールのパッチ管理システムは、パッチとサーバー設定をロールバックすることもできます。

マイクロリリースには、変更されたアーティファクトのみが含まれます。たとえば、Red Hat Single Sign-On 7.6.1 にサーバーおよび JavaScript アダプターの変更が含まれ、EAP アダプターではなく、サーバーおよび JavaScript アダプターのみがリリースされ、更新が必要になります。

1.2. KEYCLOAK から RH-SSO への移行

コミュニティープロジェクトである Keycloak から Red Hat Single Sign-On (サポートされる Red Hat 製品) に移行できます。

前提条件

- アップグレード前に新機能を確認するには、変更 を確認します。
- 正しいバージョンの Keycloak が開始点としてインストールされていることを確認します。Red Hat Single Sign-On 7.6 に移行するには、まずKeycloak 18.0.0 をインストールします。

手順

1. 「マイナーアップグレード」 の手順を実行します。この手順には **マイナーアップグレード** のラベルが付けられていますが、この移行には同じ手順が適用されます。
2. 「アダプターアップグレード手順」を実行します。

第2章 リリース固有の変更

アップグレードする前に、これらの変更を慎重に確認してください。

2.1. RH SSO 7.6

Red Hat Single Sign-On 7.5 から Red Hat Single Sign-On 7.6 に以下の変更が加えられました。

2.1.1. 段階的な認証

段階的な認証は新しい機能です。この機能は、トークンに **acr** 要求を追加する必要があるプロトコルマッパーが含まれる **acr** クライアントスコープを提供します。**acr** 要求は、このバージョンよりも前のように自動的に追加されなくなりましたが、このクライアントスコープおよびプロトコルマッパーを使用して追加されます。

クライアントスコープはレルムの「デフォルト」クライアントスコープとして追加されるため、新規に作成されるすべてのクライアントに追加されます。パフォーマンス上の理由から、クライアントスコープは移行時に既存のすべてのクライアントに自動的に追加されません。移行後に、クライアントにはデフォルトで **acr** 要求がありません。以下のアクションの実行について検討してください。

- 段階的な認証機能を使用する予定がなく、トークンの **acr** 要求に依存する場合は、Server Installation and Configuration Guide で説明されているように **step_up_authentication** 機能を無効にできます。要求は、通常の認証の場合は **1** の値で、SSO 認証の場合は **0** の値で、それぞれ追加されます。

- 管理 REST API または管理コンソールにより、手動でクライアントに **acr** クライアントスコープを追加します。これは、特に段階的な認証を使用する場合は必要です。レルムに多数のクライアントがあり、それらすべてに **acr** 要求を使用する場合は、データベースに対してこれと同様の SQL をトリガーできます。ただし、Red Hat Single Sign-On がすでに起動している場合は、キャッシュをクリアするか、サーバーを再起動することを忘れないようにしてください。

```
insert into CLIENT_SCOPE_CLIENT (CLIENT_ID, SCOPE_ID, DEFAULT_SCOPE) select
CLIENT.ID as CLIENT_ID, CLIENT_SCOPE.ID as SCOPE_ID, true as DEFAULT_SCOPE
from CLIENT_SCOPE, CLIENT where CLIENT_SCOPE.REALM_ID='test' and
CLIENT_SCOPE.NAME='acr' and CLIENT.REALM_ID='test' and CLIENT.PROTOCOL='openid-
connect';
```

2.1.2. OpenID Connect のログアウト

以前のバージョンの Red Hat Single Sign-On は、**http(s)://example-host/auth/realms/my-realm-name/protocol/openid-connect/logout?redirect_uri=encodedRedirectUri** などのログアウトエンドポイント URL を開いて、ユーザーの自動ログアウトおよびアプリケーションへのリダイレクトをサポートしていました。この実装は使い易かった一方、パフォーマンスおよびセキュリティーに悪影響を及ぼす可能性がありました。新しいバージョンでは、OpenID Connect RP-Initiated Logout 仕様に基づくログアウトへの対応が改善されました。パラメーター **redirect_uri** はサポートされなくなりました。また、新しいバージョンでは、ユーザーがログアウトを確認する必要があります。ログインに使用する ID トークンで **id_token_hint** パラメーターと共に **post_logout_redirect_uri** パラメーターを追加すると、確認を省略し、アプリケーションに自動的にリダイレクトすることができます。

既存のデプロイメントは以下のように影響を受けます。

- アプリケーションが **redirect_uri** パラメーターで直接リンクを使用してエンドポイントをログアウトする場合、上記のようにこれを変更する必要がある場合があります。**redirect_uri** パラメーターを完全に削除するか、**id_token_hint** パラメーターおよび **post_logout_redirect_uri**

パラメーターに置き換えることを検討してください。

- java アダプターを使用し、アプリケーションが **httpServletRequest.logout（）** を呼び出してログアウトする場合、この呼び出しはログアウトエンドポイントのバックチャネルバリアントを使用し、これは変更されないため、影響を受けません。

- 最新の javascript アダプターを使用する場合も、影響を受けません。ただし、アプリケーションが古いバージョンの JavaScript アダプターを使用する場合、このアダプターは非推奨の **redirect_uri** パラメーターでログアウトエンドポイントのバリアントを使用するため、影響を受けます。この場合は、最新バージョンの JavaScript アダプターにアップグレードする必要がある場合があります。

- Node.js アダプターでは、JavaScript アダプターと同じガイドラインが適用されます。アダプターの古いバージョンは非推奨の **redirect_uri** パラメーターを使用するため、最新バージョンに更新することが推奨されます。最新の Node.js アダプターでは、ドキュメントまたは Node.js アダプターのサンプルで説明されているように **/logout** URL に基づくログアウトを使用する限り、影響を受けません。ただし、アプリケーションがメソッド **keycloak.logoutUrl** を直接使用する場合は、このメソッドの 2 つ目の引数として **idTokenHint** を追加することを検討してください。このバージョンで、2 つ目の引数として **idTokenHint** を追加する可能性が新たに追加されました。**idTokenHint** は、ログイン時に取得した有効な ID トークンである必要があります。**idTokenHint** の追加はオプションですが、これを省略する場合は、ユーザーは前述のようにログアウト画面を確認する必要があります。また、ログアウト後はアプリケーションにリダイレクトされません。

アプリケーションが古い形式の **redirect_uri** パラメーターを使用できる後方互換性のオプションがあります。

standalone-*.xml ファイルに以下の設定を含めると、このパラメーターを有効にすることができます。

```
<spi name="login-protocol">
    <provider name="openid-connect" enabled="true">
        <properties>
            <property name="legacy-logout-redirect-uri" value="true"/>
        </properties>
    </provider>
</spi>
```

この設定では、**redirect_uri** パラメーターの形式を引き続き使用できます。**id_token_hint** が省略されている場合は、確認画面が必要なことに注意してください。

警告

後方互換性のスイッチは、今後のバージョンで削除される予定です。このスイッチを使用せずに、上述のとおりできるだけ早くクライアントを更新することが推奨されます。

2.1.3. upload-scripts 機能の削除

以前のバージョンの Red Hat Single Sign-On は、管理コンソールや REST API などの管理インターフェースを介して JavaScript コードの管理をサポートしていました。このバージョン以降はこれは不可能になり、以下のプロバイダーを設定するためにスクリプトをサーバーにデプロイしなければならな

くなりました。

- OpenID Connect Script Mapper
- Script Authenticator (認証の実行)
- JavaScript Policies

サーバーにスクリプトをデプロイする方法は、ドキュメントを参照してください。スクリプトを使用するには、テクノロジープレビュー機能の**scripts**を有効にする必要がある点に注意してください。

```
./standalone.sh -Dkeycloak.profile=preview
```

スクリプトをデプロイする際に、サーバーは対応するプロバイダーを自動的に作成し、認証フロー、マッパー、および認可ポリシーの設定時に選択できるようにします。

通常、レルムを更新する手順は次のとおりです。

- アップグレードする前に、使用しているスクリプトプロバイダーを削除します。
- アップグレード後に、ドキュメント の手順に従ってスクリプトをデプロイします。
- サーバーにデプロイされたスクリプトから作成されたプロバイダーを使用するように、認証フロー、マッパー、およびクライアント認証設定を更新します。

2.1.4. アカウントコンソールのPatternflyのアップグレード

Patternfly (PF) React ライブラリーが更新され、**@patternfly/react-core** が v3.153.3 から v4.147.0、**@patternfly/react-icons**がv3.15.16 から v 4.11.8 に、 **@patternfly/react-styles** が v3.7.14 から v4.11.8 に更新されました。アカウントコンソールを PF 設計標準に合わせるために、マイナーな UI 更新が加えられました。

PF の重大な変更により、カスタム開発アカウント UI は、これらの更新と互換性がない場合があります。PF コンポーネントに対する提案の更新により、ほとんどの重大な変更が解決可能です。

リソース：

- [Patternfly ドキュメント](https://www.patternfly.org)

重大な変更が含まれることがわかっているコンポーネント：

- Alert
- **action** の提案が **actionClose**に変更
- Expandable
- **ExpandableSection**に名称変更
- Title
- size 属性が **TitleSizes**を使用するようになりました
- DataListContent
- **noPadding** が **hasNoPadding**に変更

- Grid、Stack、Level、Gallery
- **gutter** 属性が **hasGutter**に変更
- Modal
- サイジング制御が**isLarge** (例)から**ModalVariant**の使用に変更(例:**variant= {ModalVariant.large}**)
- Select
- **ariaLabelTypeAhead** が **typeAheadAriaLabel**に変更
- **isExpanded** が **isOpen**に変更
- **ariaLabelledBy** が **aria-labelledby**に変更
- DataListContent
- **noPadding** が **hasNoPadding**に変更

2.1.5. クライアントポリシーの移行：client-scopes

client-scopes条件を含むポリシーを使用し、JSON ドキュメントを直接編集した場合は、JSON ドキュメントの「scope」フィールド名を「scopes」に変更する必要があります。

2.1.6. Liquibase をバージョン 4.6.2 にアップグレード

Liquibase はバージョン 3.5.5 から 4.6.2 に更新されました。これには、いくつかのバグ修正や、**ServiceLoader** を使用してカスタム拡張機能を登録する新しい方法などが含まれます。

アップグレードガイド(特に、**アップグレード前の既存データベースのバックアップ**)に厳密に従います。最善の努力でLiquibase のアップグレードの結果をテストしましたが、一部のインストールで不明な特定のセットアップを使用している可能性があります。

2.1.7. Red Hat Single Sign-On Operator の非推奨の機能

今回のリリースにより、Red Hat Single Sign-On Operator の Keycloak CR で **podDisruptionBudget** フィールドが非推奨になりました。このオプションフィールドは、Operator が OCP 4.12 以降のバージョンにデプロイされると無視されます。

回避策として、以下のようにクラスターで Pod Disruption Budget を手動で作成できます。

```
apiVersion: policy/v1
kind: PodDisruptionBudget
metadata:
  labels:
    app: keycloak
  name: keycloak
spec:
  maxUnavailable: 1
  selector:
    matchLabels:
      component: keycloak
```

Kubernetes のドキュメント も参照してください。

2.2. RH SSO 7.5

Red Hat Single Sign-On 7.4 から Red Hat Single Sign-On 7.5 に以下の変更が加えられました。

2.2.1. EAP 7.4 へのアップグレード

Red Hat Single Sign-On サーバーが EAP 7.4 を基礎となるコンテナーとして使用するようにアップグレードされました。この変更には、特定の Red Hat Red Hat Single Sign-On 機能は直接含まれていませんが、移行に関連する変更がいくつかあります。

2.2.1.1. 依存関係の更新

この依存関係は、EAP 7.4 サーバーが使用するバージョンに更新されました。たとえば、Infinispan コンポーネントバージョンは 11.0 になりました。

2.2.1.2. 設定変更

standalone(-ha).xml ファイルおよび domain.xml ファイルにはいくつかの設定変更があります。設定ファイルの自動移行を処理するには、「Red Hat Single Sign-On サーバーのアップグレード」 セクションを参照してください。

2.2.1.3. SmallRye の手動変更

standalone.xml に SmallRye モジュールへの参照が含まれる場合は、手動の変更が必要になります。これらのモジュールは基礎となる JBoss EAP ディストリビューションから削除され、設定がそれらを参照するとサーバーは起動しません。**migrate-standalone.cli** によるサーバー設定の移行は、設定に変更を加える前に失敗します。

この問題を修正するには、SmallRye モジュールを参照するすべての行を削除します。デフォルト設定では、特に以下の行を削除する必要があります。

```
<extension module="org.wildfly.extension.microprofile.config-smallrye"/>
<extension module="org.wildfly.extension.microprofile.health-smallrye"/>
<extension module="org.wildfly.extension.microprofile.metrics-smallrye"/>
```

```
<subsystem xmlns="urn:wildfly:microprofile-config-smallrye:1.0"/>
<subsystem xmlns="urn:wildfly:microprofile-health-smallrye:2.0" security-enabled="false" empty-liveness-checks-status="${env.MP_HEALTH_EMPTY_LIVENESS_CHECKS_STATUS:UP}" empty-readiness-checks-status="${env.MP_HEALTH_EMPTY_READINESS_CHECKS_STATUS:UP}"/>
<subsystem xmlns="urn:wildfly:microprofile-metrics-smallrye:2.0" security-enabled="false" exposed-subsystems="*" prefix="${wildfly.metrics.prefix:wildfly}"/>
```

2.2.1.4. データセンター間のレプリケーションの変更

- RHDG サーバーをバージョン 8.x にアップグレードする必要があります。古いバージョンはまだ機能する可能性はありますが、テストされなくなったため保証されていません。
- Infinispan キャッシュの設定時に remote-store 要素に追加された **protocolVersion** プロパティーを使用することが推奨されます。RHDG サーバー 8.x に接続する場合は、hotrod プロトコルバージョンの hotrod は 2.9 です。Infinispan ライブラリーバージョンは、Red Hat Single

Sign-On サーバーおよび RHDG サーバーごとに異なります。詳細は、Cross-Datacenter のドキュメントを参照してください。

- **connectioninfinispan** サブシステムの下に **remoteStoreSecurityEnabled** プロパティーを使用することが推奨されます。詳細は、Cross-Datacenter のドキュメントを参照してください。

2.2.2. UserModel の移行

UserModel には、カスタム属性に変換される特定のフィールド (**username**、**email**、**firstName**、および **lastName**) が含まれます。この変更により、今後のバージョンで Red Hat Single Sign-On により高度なユーザープロファイルを追加する準備が加えられました。

注記

正確な名前のカスタム属性を持つユーザーが含まれる場合、これらの属性はデータベースから読み取ることはなくなり、削除される可能性があります。したがって、RH SSO 7.5 にアップグレードする前に、これらの名前のいずれかに一致するカスタム属性の名前を変更します。

この状況は、**username** も **UserModel.getFirstAttribute(UserModel.USERNAME)** によってアクセス可能であることを意味しています。他のフィールドには同様の影響が存在します。**UserModel**を直接的または間接的にサブクラス化するSPIの実装者は、**setUsername**と**setSingleAttribute(UserModel.USERNAME, ...)**（および他のフィールドについても同様）の間の動作が一貫していることを確認する必要があります。

ポリシー評価機能のユーザーは、評価する属性の数を使用する場合にはポリシーを調整する必要があります。すべてのユーザーには、デフォルトで 4 つの新しい属性が使用されます。

UserModel の公開 API は変更されませんでした。フロントエンドリソースやユーザーデータにアクセスする SPI への変更は必要ありません。また、データベースはまだ変更されませんでした。

2.2.3. PatternFly 4 へのアップグレード

Red Hat Single Sign-On ログイン用のテーマのコンポーネントは PatternFly 4 にアップグレードされました。古い PatternFly 3 は新しいパターンと同時に実行されるので、PF3 コンポーネントを保持することができます。

ただし、ログインテーマの設計にいくつかの変更が加えられました。カスタムログインテーマを新しいバージョンにアップグレードしてください。必要な変更に関する例は、**examples/themes/theme/sunrise** ディレクトリーにあります。追加の設定は必要ありません。

2.2.4. Instagram IdP の新しい API

Instagram IdP は新しい API を使用するようになりました。古いレガシー API は**非推奨**となりました。この変更には、新しい API クレデンシャルが必要です。詳細は、Server Administration Guide を参照してください。

Instagram IdP を使用して Instagram にログインするユーザーの場合は、パスワードなどの別の認証方法が必要になります。ログインして、Instagram ソーシャルリンクを手動で更新したり、Red Hat Single Sign-On で新しいアカウントを作成したりすることができます。この制限は、以前の API の Instagram ユーザー ID は新しい API と互換性がないためです。ただし、新規 API は一時的に新規ユーザー ID と古いユーザー ID の両方を返し、移行を許可します。Red Hat Single Sign-On は、ユーザーのログイン時に ID を自動的に移行します。

2.2.5. SSRF 保護の有効な要求 URI

OpenID Connect パラメーターの **request_uri** を使用する場合、クライアントが SSRF 攻撃から保護するように **有効なリクエスト URI** を設定する必要があります。

この機能は、クライアントの詳細ページの管理コンソール、または管理 REST API またはクライアント登録 API で設定できます。有効なRequest URI には、特定のクライアントで許可される Request URI 値の一覧が含まれている必要があります。

代わりに、**Valid Redirect URIs** オプションなどのワイルドカードまたは相対パスを使用することもできます。ただし、セキュリティー上の理由から、できるだけ特定の値を使用することが推奨されます。

2.2.6. 読み取り専用ユーザー属性

読み取り専用ユーザー属性を使用できるようになりました。これらのユーザー属性の一部は、REST API または Red Hat Single Sign-On ユーザーインターフェースを使用してユーザーを更新する場合、ユーザーまたは管理者が編集することはできません。特に、この変更は以下のいずれかを使用する際に重要です。

- カスタムユーザーストレージプロバイダー
- カスタムオーセンティケーター
- ユーザー属性を基に承認を確立するカスタムの JavaScript 認証ポリシー
- X.509 証明書をユーザー ID にマッピングするカスタム属性で X.509 オーセンティケーター
- ユーザー属性の一部が、単純なユーザープロファイル情報ではなく、認証/承認/アイデンティティーコンテキストを保存するメタデータとして使用されるその他のカスタム機能。

詳細は、Threat model mitigation chapter を参照してください。

2.2.7. Docker 認証の後にユーザーセッションは必要なし

Docker プロトコルを使用した認証に成功すると、ユーザーセッションは作成されません。詳細は、Server Administration Guide を参照してください。

2.2.8. デフォルトでは、更新トークンなしでクライアント認証情報が付与

この Red Hat Single Sign-On バージョンの場合、OAuth2 Client Credentials Grant エンドポイントはデフォルトでトークンの更新を実行しません。この動作は、OAuth2 仕様に合わせて調整されます。

そのため、クライアント認証情報の認証に成功した後に、Red Hat Single Sign-On サーバー側でユーザーセッションが作成されません。その結果、パフォーマンスとメモリーの消費が向上されます。Client Credentials Grant を使用するクライアントでは、更新トークンの使用を停止することが推奨されます。代わりに、**refresh_token** を付与タイプとして使用する代わりに、**grant_type=client_credentials** のすべての要求で認証することが推奨されます。

これに関連して、Red Hat Single Sign-On は OAuth2 Revocation Endpoint のアクセストークンの取り消しをサポートします。そのため、必要に応じて、クライアントは個別のアクセストークンを取り消すことができます。

後方互換性を維持するには、以前のバージョンの動作を維持することができます。このアプローチが使用されると、クライアント認証情報の付与を使用した認証に成功した後に更新トークンが発行されます。また、ユーザーセッションが作成されます。特定のクライアントでは、以下のように管理コンソールで以前の動作を有効にすることができます。

手順

1. メニューで **Clients** をクリックします。
2. 変更するクライアントをクリックします。
3. **OpenID Connect Compatibility Modes** セクションを展開します。
4. **Use Refresh Tokens For Client Credentials Grant**を **ON** に切り替えます。
5. **Save** をクリックします。

2.2.9. 標準以外のトークンイントロスペクションエンドポイントが削除される

以前のバージョンでは、Red Hat Single Sign-On がアドバタイズされた 2 つのイントロスペクションエンドポイント（**token_introspection_endpoint** および **introspection_endpoint**）後者は RFC-8414 で定義されるものと同じです。前者は非推奨となり、削除されました。

2.2.10. LDAP インポートなしの修正

以前の Red Hat Single Sign-On バージョンでは、LDAP プロバイダーが **Import Users** OFF で設定されている場合、LDAP 以外のマッピング属性が変更されてもユーザーを更新することができました。この状況では、混乱を生じさせる動作が発生していました。更新されると表示される属性ですが、更新されませんでした。

たとえば、管理 REST API でユーザーを更新しようとし、ユーザーが間違った属性の変更があった場合、更新が可能でした。現在のバージョンでは更新は不可能であり、理由について即座に通知されます。

2.3. RH SSO 7.4

Red Hat Single Sign-On 7.3 から Red Hat Single Sign-On 7.4 に以下の変更が加えられました。

2.3.1. EAP 7.3 へのアップグレード

Red Hat Single Sign-On サーバーが EAP 7.3 を基礎となるコンテナーとして使用するようにアップグレードされました。この変更には、特定の Red Hat Red Hat Single Sign-On 機能は直接含まれていませんが、移行に関連する変更がいくつかあります。

2.3.1.1. 依存関係の更新

依存関係は、EAP 7.3 サーバーが使用するバージョンに更新されました。たとえば、Infinispan コンポーネントバージョンは 9.3.1.Final になりました。

2.3.1.2. 設定変更

standalone(-ha).xml ファイルおよび domain.xml ファイルにはいくつかの設定変更があります。「Red Hat Single Sign-On サーバーのアップグレード」セクションに従って、設定ファイルの自動移行を処理します。

2.3.1.3. データセンター間のレプリケーションの変更

RHDG をバージョン 7.3 にアップグレードする必要があります。古いバージョンはまだ機能する可能性はありますが、テストされていないため、機能する保証はありません。

2.3.2. 認証フローの変更

認証フローに関連するリファクタリングおよび改善がありました。これには、移行時に注意する必要があります。

2.3.2.1. REQUIRED と ALTERNATIVE の実行は、同じ認証フローでは対応していません。

以前のバージョンでは、同じレベルの同じ認証フローに REQUIRED および ALTERNATIVE の実行を行うことができました。このアプローチにはいくつかの問題があり、Authentication SPI のリファクタリングが行われました。つまり、これは有効ではなくなります。ALTERNATIVE および REQUIRED の実行が同じレベルで設定されている場合、ALTERNATIVE 実行は無効にされます。

したがって、このバージョンに移行すると、既存の認証フローが移行されますが、以前のバージョンの動作が保持されます。認証フローに ALTERNATIVE 実行が REQUIRED 実行と同じレベルに含まれる場合は、ALTERNATIVE 実行が別個の REQUIRED サブフローに追加されます。

このストラテジーでは、各認証フローで以前のバージョンと同じまたは同様の動作が保証されます。ただし、認証フローの構成を確認し、期待どおりに機能することを再確認することはできます。この推奨事項は、カスタムオーセンティケーターの実装を使用したカスタマイズされた認証フローに特に適用されます。

2.3.2.2. OPTIONAL の実行要件が削除されました

移行に関して最も重要な変更は、認証の実行から OPTIONAL 要件のサポートを削除し、それを CONDITIONAL 要件に置き換えることです。これにより、柔軟性が向上します。

以前のバージョンで設定された OPTIONAL オーセンティケーターは、CONDITIONAL サブフローに置き換えられます。これらのサブフローには、Condition - User 構成条件が最初の実行として構成され、以前の OPTIONAL オーセンティケーター (OTP フォームなど) が 2 番目の実行として構成されています。ユーザーの場合には、認証中の動作は、以前のバージョンの動作と一致します。

2.3.2.3. SPI の変更点

Java Authentication SPI および Credential Provider SPI にはいくつかの変更が存在します。

インターフェースオーセンティケーターは変更されていませんが、新しい認証情報タイプ (CredentialModel のサブクラス) を導入する高度なオーセンティケーターを開発する場合は、影響を受ける可能性があります。CredentialProvider インターフェースには変更があり、CredentialValidator などの新しいインターフェースが導入されています。

また、オーセンティケーターが OPTIONAL 実行要件に対応している場合は、影響を受ける可能性があります。詳細については、サーバー開発ガイドの最新の認証例を再確認することが推奨されます。

2.3.2.4. Freemarker テンプレートの変更

フリーマーカーテンプレートに変更があります。ログインフォームまたは一部のアカウントフォーム、特に OTP に関連するフォーム用のカスタムフリーマーカーテンプレートを使用した独自のテーマがある場合は、影響を受ける可能性があります。このバージョンの FreeMarker テンプレートの変更を確認し、テンプレートを調整することが推奨されます。

2.3.3. 重複した最上位のグループ

本リリースでは、レルムに重複した最上位グループを作成できる問題を修正しています。以前の重複グループが存在すると、アップグレードプロセスが失敗します。Red Hat Single Sign-On サーバーは、H2、MariaDB、MySQL、または PostgreSQL データベースを使用している場合は、この問題の影響を

受けます。アップグレードを開始する前に、サーバーに重複した最上位グループが含まれているかどうかを確認します。たとえば、以下の SQL クエリーはデータベースレベルで実行して一覧表示できます。

```
SELECT REALM_ID, NAME, COUNT(*) FROM KEYCLOAK_GROUP WHERE PARENT_GROUP is NULL GROUP BY REALM_ID, NAME HAVING COUNT(*) > 1;
```

同じ名前のレルムごとに 1 つの最上位グループのみが存在します。重複は、アップグレード前に確認および削除する必要があります。アップグレードのエラーには、**Change Set META-INF/jpa-changelog-9.0.1.xml::9.0.1- KEYCLOAK-12579-add-not-null-constraint::keycloak failed** というメッセージが含まれます。

2.3.4. ユーザー認証情報の変更

ユーザー認証情報の保存に柔軟性が追加されました。一方で、すべてのユーザーが、複数の OTP クレデンシャルなど、同じタイプの複数の認証情報を持つことができます。これに関連してデータベーススキーマに変更がいくつか存在しますが、前のバージョンの認証情報は新しい形式に更新されます。ユーザーは、以前のバージョンで定義されたパスワードまたは OTP 認証情報を使用してログインできます。

2.3.5. 新しい任意のクライアントスコープ

MicroProfile/JWT Auth Specification で定義される要求を処理するために microprofile-jwt の任意のクライアントスコープが追加されました。この新しいクライアントスコープは、認証済みユーザーのユーザー名を upn 要求に設定し、レルムロールを groups 要求に設定するプロトコルマッパーを定義します。

2.3.6. ユーザーロケールの処理が改善

ログインページのロケールの選択方法や、ユーザーのロケールが更新された時などに多くの改良が加えられました。詳細は、Server Administration Guide を参照してください。

2.3.7. JavaScript アダプターのレガシープロミス

JavaScript アダプターに promiseType を設定する必要はなく、いずれも同時に利用できるようになりました。レガシー API (success および error) がある時点ですぐにネイティブな promise API (then および catch) を使用するようにアプリケーションを更新することが推奨されます。

2.3.8. サーバーへのスクリプトのデプロイ

これまで、管理者は Red Hat Single Sign-On 管理コンソールおよび RESTful Admin API を使用してスクリプトをサーバーにアップロードできるようになりました。この機能は無効にされています。ユーザーはスクリプトを直接サーバーにデプロイする必要があります。詳細は、JavaScript Providers を参照してください。

2.3.9. JavaScript アダプターのクライアント認証情報

以前のリリースでは、開発者は JavaScript アダプターにクライアント認証情報を提供することができました。クライアント側のアプリは秘密を守るという点で安全ではないため、現在、この機能は削除されました。prompt=none をデフォルトの IDP に伝播する機能

クライアントからの Accepts prompt=none forward という名前の OIDC アイデンティティープロバイダー設定にスイッチを追加し、prompt=none クエリーパラメーターを含む転送されたリクエストを処理できる IDP を特定します。

これまで、prompt=none で認証要求を受け取ると、ユーザーが IDP によって認証されているかどうかを確認せずに、ユーザーがレルムで認証されていない場合、レルムは login_required エラーを返していました。今後、認証要求に対してデフォルトの IDP を決定できる場合 (kc_idp_hint クエリーパラメータを使用するか、レルムのデフォルトの IDP を設定することにより) と、クライアントスイッチからの Accepts prompt=none 転送が IDP に対して有効になっている場合、認証要求は IDP に転送され、ユーザーが IDP で認証されているかどうかが確認されます。

この切り替えは、デフォルトの IDP が指定されている場合にのみ考慮されることに注意してください。この場合は、ユーザーに IDP の選択を求めることなく、認証要求を転送する場所がわかります。デフォルトの IDP を決定できない場合は、認証要求を実行するためにどちらが使用されるかを想定できないため、要求の転送は実行されません。

2.3.10. 新しいデフォルトホスト名プロバイダー

要求および固定ホスト名プロバイダーが新規のデフォルトのホスト名プロバイダーに置き換えられました。要求および固定のホスト名プロバイダーは非推奨となり、できるだけ早くデフォルトのホスト名プロバイダーに切り替えることが推奨されます。

2.3.11. 非推奨または削除された機能

特定の機能のステータスが変更になりました。

2.3.11.1. トークン表現の Java クラスの非推奨のメソッド

2038 年に、int は1970年以降、秒の値を保持できなくなります。そのため、これらを長い値に更新する作業を行っています。トークン表現では、さらに問題があります。int は、デフォルトでは JSON 表現で 0 になりますが、含めるべきではありません。

非推奨となった正確な方法や代替方法の詳細は、JavaDocs ドキュメント を参照してください。

2.3.11.2. スクリプトのアップロード

管理 REST エンドポイントおよびコンソールを使用したスクリプトのアップロードが非推奨となりました。これは今後のリリースで削除されます。

2.3.12. 承認サービスの Drools ポリシー

承認サービスの Drools ポリシーが削除されました。

2.4. RH-SSO 7.3

RH-SSO 7.2 から RH-SSO 7.3 に以下の変更が加えられました。

2.4.1. 承認サービスの変更

UMA 2.0 のサポートが追加されました。このバージョンの UMA 仕様では、パーミッションの取得方法に関する重要な変更がいくつか導入されました。

以下は、UMA 2.0 サポートによって導入される主な変更です。詳細は、Authorization Services Guide を参照してください。

承認 API が削除されました。

UMA 2.0 (UMA 1.0) より前のバージョンでは、クライアントアプリケーションは Authorization API

を使用して RPT の形式でサーバーからパーミッションを取得していました。新しいバージョンの UMA 仕様では、Red Hat Single Sign-On からも削除された Authorization API が削除されました。UMA 2.0 では、特定の付与タイプを使用して、RPT がトークンエンドポイントから取得できるようになりました。詳細は、Authorization Services Guideを参照してください。

エンタイトルメント API が削除されました。

UMA 2.0 の導入に伴い、トークンエンドポイントと UMA 付与タイプを活用して、Red Hat Single Sign-On から RPT を取得できるようにし、異なる API を使用しないようにすることにしました。Entitlement API によって提供される機能は同じままであり、リソースまたはスコープが提供されていない場合でも、サーバーから 1 つ以上のリソースとスコープのセットに対するアクセス許可、またはすべてのアクセス許可を取得できます。詳細は、Authorization Services Guideを参照してください。

UMA 検出エンドポイントへの変更

UMA 検出ドキュメントが変更されました。詳細は、Authorization Services Guideを参照してください。

Red Hat Single Sign-On Authorization JavaScript アダプターの変更点

Red Hat Single Sign-On Authorization JavaScript アダプター (keycloak-authz.js) は、以前と同じ動作を維持しながら、UMA 2.0 によって導入された変更に準拠するために変更されました。主な変更点は、**authorization** と **entitlement** メソッドの両方を呼び出す方法にあります。これにより、認証要求を表す特定のオブジェクトタイプが期待されます。この新しいオブジェクトタイプでは、UMA 付与タイプでサポートされる異なるパラメーターをサポートし、サーバーから取得できるパーミッションを柔軟に提供できます。詳細は、Authorization Services Guideを参照してください。

> One of the main changes introduced by this release is that you are no longer required to exchange access tokens with RPTs in
> order to access resources protected by a resource server (when not using UMA). Depending on how the policy enforcer is configured on the resource server side, you can just send regular access tokens as a bearer token and permissions will still be enforced.

Red Hat Single Sign-On Authorization Client Java API への変更

Red Hat Single Sign-On Authorization Client Java API の新しいバージョンにアップグレードする際に、一部の表現クラスが **org.keycloak:keycloak-core** の別のパッケージに移動したことに注意してください。

2.4.2. クライアントスコープに変更になったクライアントテンプレート

Client Scopes がサポートされるようになりました。これには、移行時に注意する必要があります。

クライアントスコープに変更になったクライアントテンプレート

クライアントテンプレートはクライアントスコープに変更されました。クライアントテンプレートがある場合、プロトコルマッパーおよびロールスコープのマッピングは保持されます。

名前で置き換えられたスペース

名前に空白文字が含まれるクライアントテンプレートは、クライアントスコープの名前に空白を使用できないため、空白をアンダースコアに置き換えるように名前が変更されました。たとえば、クライアントテンプレート **my template** はクライアントスコープ **my_template** に変更されます。

クライアントスコープのクライアントへのリンク

クライアントテンプレートを持つクライアントでは、対応するクライアントスコープが **Default Client Scope** としてクライアントに追加されます。そのため、プロトコルマッパーとロールマッピングはクライアントに保存されます。

レルムのデフォルトのクライアントスコープが既存のクライアントにリンクされていない

移行時に、組み込みクライアントスコープのリストが、各レルムと、**レルムのデフォルトクライア**

ントスコープ のリストが追加されます。ただし、既存のクライアントはアップグレードされず、新しいクライアントスコープは自動的に追加されません。また、プロトコルマッパーとロールスコープのマッピングはすべて既存のクライアントに保持されます。新しいバージョンでは、新規クライアントの作成時に、Realm Default Client Scopes が自動的に割り当てられ、プロトコルマッパーが割り当てられません。たとえば、クライアントのプロトコルマッパーのカスタマイズを適切に検出することは不可能であるため、移行中に既存のクライアントを変更しませんでした。既存のクライアントを更新する (プロトコルマッパーをクライアントから削除し、クライアントスコープにリンクする) 場合は、手動で行う必要があります。

競合を再度確認する必要がある

クライアントスコープの変更では、連携のリファクタリングが必要でした。これで、ロールまたはプロトコルマッパーではなく、クライアントスコープを参照するようになりました。この変更により、以前に確認されたユーザーによる永続的な同意は無効になり、ユーザーは移行後に同意ページを再度確認する必要があります。

いくつかの設定スイッチが削除される

スイッチ **Scope Param Required** がロールの詳細から削除されました。**Consent Required** スイッチおよび **Consent Text** スイッチが、プロトコルマッパーの詳細から削除されました。これらのスイッチは Client Scope 機能に置き換えられました。

2.4.3. 新しいデフォルトのクライアントスコープ

新しいレルムのデフォルトクライアントスコープ **roles** および **web-origins** を追加しました。これらのクライアントスコープには、ユーザーのロールと許可された Web オリジンをトークンに追加するプロトコルマッパーが含まれます。移行時に、これらのクライアントスコープをデフォルトのクライアントスコープとしてすべての OpenID Connect クライアントに自動的に追加する必要があります。したがって、データベースの移行が完了したら、設定は必要ありません。

2.4.3.1. プロトコルマッパー SPI の追加

これに関連して、(サポートされていない) プProtocol Mappers SPI に小規模な追加があります。カスタム ProtocolMapper を実装している場合にのみ、影響を受ける可能性があります。ProtocolMapper インターフェースには、新しい **getPriority()** メソッドがあります。メソッドでは、デフォルトの実装は 0 を返すように設定されます。プロトコルマッパー実装が、**realmAccess** プロパティーまたは **resourceAccess** プロパティーのロールに依存する場合は、マッパーの優先度を増やす必要がある場合があります。

2.4.3.2. オーディエンスの解決

認証されたユーザーには、トークンに最低でも 1 つのクライアントロールがあるすべてのクライアントのオーディエンスが、アクセストークンの **aud** 要求に自動的に追加されるようになりました。一方、アクセストークンには、それが発行されたフロントエンドクライアントのオーディエンスが自動的に含まれない場合があります。詳細は、Server Administration Guide を参照してください。

2.4.4. EAP 7.2 へのアップグレード

Red Hat Single Sign-On サーバーが EAP 7.2 を基礎となるコンテナーとして使用するようにアップグレードされました。これには、特定の Red Hat Single Sign-On サーバー機能は直接関係しませんが、言及する価値がある移行に関連する変更はほとんどありません。

依存関係の更新

この依存関係は、EAP 7.2 サーバーが使用するバージョンに更新されました。たとえば、Infinispan は 9.3.1.Final になりました。

設定変更

standalone(-ha).xml および **domain.xml** ファイルの設定変更はほとんどありません。設定ファイルの自動移行を処理するには、「Red Hat Single Sign-On サーバーのアップグレード」 セクションを参照してください。

データセンター間のレプリケーションの変更

- RHDG サーバーをバージョン 7.3 にアップグレードする必要があります。古いバージョンはまだ動作する場合もありますが、テストしていないため保証はありません。
- Red Hat Single Sign-On 設定の **remote-store** 要素の設定に、値が **2.6** の **protocolVersion** プロパティーを追加する必要があります。これは、RHD 7.3 で使用されるバージョンと互換性を持たせるために HotRod プロトコルのバージョンをダウングレードする必要があるため必要です。

2.4.5. ホスト名の設定

以前のバージョンでは、許可されたホスト名を指定するためにフィルターを使用することが推奨されます。固定ホスト名を設定することで、有効なホスト名が使用されることを確認し、内部アプリケーションが内部 IP アドレスなどの別の URL を使用して Red Hat Single Sign-On を呼び出すことができるようになっています。実稼働環境で、このアプローチに切り替えることが推奨されます。

2.4.6. JavaScript アダプターのpromise

JavaScript アダプターでネイティブの JavaScript Promise を使用できるようにするには、init オプションで、**promiseType** を **native** に設定する必要があります。

過去にネイティブな promise が利用できる場合は、従来の Keycloak の promise とネイティブの promise の両方を提供していたラッパーが返されていました。これにより、エラーハンドラーがネイティブエラーイベントの前に常に設定されていなかったため、問題が生じていました。その結果、**Uncaught (in promise)** エラーが発生していました。

2.4.7. Microsoft Identity Provider が Microsoft Graph API を使用するように更新

承認の Live SDK エンドポイントに依存してユーザープロファイルを取得するために使用される Red Hat Single Sign-On の Microsoft Identity Provider 実装。2018 年 11 月以降、Microsoft は新しい Microsoft Graph API を優先して、ライブ SDK API のサポートを削除しています。Red Hat Single Sign-On ID プロバイダーが新しいエンドポイントを使用するように更新されました。そのため、この統合が使用されている場合は、最新の Red Hat Single Sign-On バージョンにアップグレードしてください。

アプリケーションの ID 形式の変更により、「Live SDK applications」で登録されたレガシークライアントアプリケーションは Microsoft Graph エンドポイントでは動作しません。アプリケーション識別子がディレクトリーで見つからないというエラーが発生した場合は、新しいアプリケーション ID を取得するために、Microsoft Application Registration ポータルでクライアントアプリケーションを再度登録する必要があります。

2.4.8. Google ID プロバイダーが Google Sign-in 認証システムを使用するように更新されました。

承認の Google+ API エンドポイントに依存してユーザープロファイルを取得するために使用される Red Hat Single Sign-On の Google Identity Provider 実装。2019 年 3 月以降、Google は新しい Google サインイン認証システムを優先し、Google+ API のサポートは修了しています。Red Hat Single Sign-On ID プロバイダーが新しいエンドポイントを使用するように更新されました。そのため、この統合が使用されている場合は、最新の Red Hat Single Sign-On バージョンにアップグレードしてください。

アプリケーション ID がディレクトリーで見つからないというエラーが発生した場合は、クライアントアプリケーションを Google API Console ポータルに登録し、新規アプリケーション ID およびシークレットを取得する必要があります。

Google+ ユーザー情報エンドポイントで提供される標準以外の要求のカスタムマッパーを調整し、Google Sign-in API によって異なる名前に基づいて提供されることがあります。利用可能な要求に関する最新情報は、Google ドキュメントを参照してください。

2.4.9. LinkedIn Social Broker が LinkedIn API のバージョン 2 に更新

LinkedIn に応じて、すべての開発者は API および OAuth 2.0 のバージョン 2.0 に移行する必要があります。そのため、LinkedIn Social Broker を更新しました。

LinkedIn API のバージョン 2 を使用してユーザーのプロファイルを取得する際に、このブローカーを使用する既存デプロイメントでエラーが発生する場合があります。このエラーは、認証プロセス中に Profile API へのアクセスまたは特定の OAuth2 スコープの要求を許可されていない可能性があるブローカーの構成に使用されるクライアントアプリケーションに付与されたアクセス許可の欠如に関連している可能性があります。

新規に作成された LinkedIn クライアントアプリケーションであっても、クライアントが OAuth2 スコープの **r_liteprofile** および **r_emailaddress** を要求でき、少なくともクライアントアプリケーションが **https://api.linkedin.com/v2/me** エンドポイントから現在のメンバーのプロファイルを取得できることを確認する必要があります。

メンバーの情報へのアクセスや現在のメンバーの Profile API によって返される要求のセットが LinkedIn によって課され、LinkedIn Social Broker はデフォルトユーザー名としてメンバーのメールアドレスを使用するようになりました。これは、認証中に承認要求を送信する際に、常に **r_emailaddress** が設定されていることを示しています。

2.5. RH-SSO 7.2

RH-SSO 7.1 から RH-SSO 7.2 に以下の変更が加えられました。

2.5.1. 新しいパスワードハッシュアルゴリズム

パスワードハッシュアルゴリズムを新たに追加しました (pbkdf2-sha256 および pbkdf2-sha512)。新しいレルムは、27500 ハッシュの反復で pbkdf2-sha256 ハッシュアルゴリズムを使用します。pbkdf2-sha256 は pbkdf2 よりも若干高速であるため、反復は 20000 から 27500 に増加しました。

パスワードポリシーにハッシュアルゴリズム (未指定) および反復 (20000) のデフォルト値が含まれる場合は、既存のレルムがアップグレードされます。ハッシュの反復を変更した場合は、よりセキュアなハッシュアルゴリズムを使用する場合は pbkdf2-sha256 を手動で変更する必要があります。

2.5.2. ID トークンには scope=openid が必要です。

RH-SSO 7.0 では、認証要求に **scope=openid** クエリーパラメーターが存在するかどうかに関わらず ID トークンが返されました。これは、OpenID Connect の仕様に従って正しくありません。

RH-SSO 7.1 では、このクエリーパラメーターをアダプターに追加しましたが、以前の動作を維持し、移行に対応しました。

RH-SSO 7.2 では、この動作が変更され、要求を OpenID Connect 要求としてマークするために **scope=openid** クエリーパラメーターが必要になりました。このクエリーパラメーターを省略すると、ID トークンは生成されません。

2.5.3. Microsoft SQL Server には追加の依存関係が必要

Microsoft JDBC Driver 6.0 では、JDBC ドライバーモジュールに追加の依存関係を追加する必要があります。Microsoft SQL Server の使用時に **NoClassDefFoundError** エラーが発生した場合は、以下の依存関係を JDBC ドライバーの **module.xml** ファイルに追加します。

```
<module name="javax.xml.bind.api"/>
```

2.5.4. OpenID Connect Authentication Response に session_state パラメーターを追加

OpenID Connect Session Management 仕様では、**session_state** パラメーターが OpenID Connect Authentication Response に存在する必要があります。

RH-SSO 7.1 ではこのパラメーターはありませんでしたが、仕様で要求されているように、Red Hat Single Sign-On はデフォルトでこのパラメーターを追加します。

ただし、OpenID Connect/OAuth2 アダプター、特に古い Red Hat Single Sign-On アダプター (RH-SSO 7.1 以前など) には、この新しいパラメーターで問題が発生する可能性があります。

たとえば、クライアントアプリケーションへの認証に成功すると、パラメーターは常にブラウザー URL に表示されます。RH-SSO 7.1、もしくはレガシー OAuth2 または OpenID Connect アダプターを使用する場合は、認証応答への **session_state** パラメーターの追加を無効にすると役に立つ場合があります。これは、「古いアダプターとの互換性」 で説明しているように、**OpenID Connect Compatibility Modes** のセクションにあるクライアントの詳細で、Red Hat Single Sign-On 管理コンソールの特定のクライアントに対して実行できます。**認証応答から除外セッションの状態** スイッチがあり、**session_state** パラメーターを認証応答に追加しないようにオンにすることができます。

2.5.5. Microsoft Identity Provider が Microsoft Graph API を使用するように更新

バージョン 7.2.4 までの Red Hat Single Sign-On での Microsoft Identity Provider の実装は、承認とユーザープロファイルの取得を Live SDK エンドポイントに依存しています。2018 年 11 月以降、Microsoft は新しい Microsoft Graph API を優先して、ライブ SDK API のサポートを削除しています。Red Hat Single Sign-On ID プロバイダーが新しいエンドポイントを使用するように更新されました。そのため、この統合が使用されている場合は、Red Hat Single Sign-On バージョン 7.2.5 以降にアップグレードしてください。

アプリケーションの ID 形式の変更により、「Live SDK applications」で登録されたレガシークライアントアプリケーションは Microsoft Graph エンドポイントでは動作しません。アプリケーション識別子がディレクトリーで見つからないというエラーが発生した場合は、新しいアプリケーション ID を取得するために、Microsoft Application Registration ポータルでクライアントアプリケーションを再度登録する必要があります。

2.5.6. Google ID プロバイダーが Google Sign-in 認証システムを使用するように更新されました。

バージョン 7.2.5 までの Red Hat Single Sign-On での Google Identity Provider の実装は、承認とユーザープロファイルの取得を Google+ API エンドポイントに依存しています。2019 年 3 月以降、Google は新しい Google サインイン認証システムを優先し、Google+ API のサポートは修了しています。Red Hat Single Sign-On ID プロバイダーが新しいエンドポイントを使用するように更新されました。そのため、この統合が使用されている場合は、Red Hat Single Sign-On バージョン 7.2.6 以降にアップグレードしてください。

アプリケーション ID がディレクトリーで見つからないというエラーが発生した場合は、クライアントアプリケーションを Google API Console ポータルに登録し、新規アプリケーション ID およびシークレットを取得する必要があります。

Google+ ユーザー情報エンドポイントで提供される標準以外の要求のカスタムマッパーを調整し、Google Sign-in API によって異なる名前に基づいて提供されることがあります。利用可能な要求に関する最新情報は、Google ドキュメントを参照してください。

2.5.7. LinkedIn Social Broker が LinkedIn API のバージョン 2 に更新

LinkedIn に応じて、すべての開発者は API および OAuth 2.0 のバージョン 2.0 に移行する必要があります。そのため、LinkedIn Social Broker を更新しました。この統合を使用している場合は、Red Hat Single Sign-On バージョン 7.2.6 以降にアップグレードしてください。

LinkedIn API のバージョン 2 を使用してユーザーのプロファイルを取得する際に、このブローカーを使用する既存デプロイメントでエラーが発生する場合があります。このエラーは、認証プロセス中に Profile API へのアクセスまたは特定の OAuth2 スコープの要求を許可されていない可能性があるブローカーの構成に使用されるクライアントアプリケーションに付与されたアクセス許可の欠如に関連している可能性があります。

新規に作成された LinkedIn クライアントアプリケーションであっても、クライアントが OAuth2 スコープの **r_liteprofile** および **r_emailaddress** を要求でき、少なくともクライアントアプリケーションが **https://api.linkedin.com/v2/me** エンドポイントから現在のメンバーのプロファイルを取得できることを確認する必要があります。

メンバーの情報へのアクセスや現在のメンバーの Profile API によって返される要求のセットが LinkedIn によって課され、LinkedIn Social Broker はデフォルトユーザー名としてメンバーのメールアドレスを使用するようになりました。これは、認証中に承認要求を送信する際に、常に **r_emailaddress** が設定されていることを示しています。

2.6. RH-SSO 7.1

RH-SSO 7.0 から RH-SSO 7.1 に以下の変更が加えられました。

2.6.1. レルムキー

RH-SSO 7.0 の場合は、1 つのキーセットのみがレルムに関連付けることができます。つまり、キーを変更すると、現在のクッキーとトークンがすべて無効になり、すべてのユーザーが再認証する必要があります。RH-SSO 7.1 で、1 つのレルムに対して複数のキーのサポートが追加されました。いつでも、1 セットのキーが署名の作成に使用されるアクティブセットですが、署名の検証に使用される複数のキーが存在する可能性があります。つまり、古い Cookie とトークンを検証してから、新しい署名で更新できることを意味し、キーを変更した時にユーザーが認証したままになることを可能にあります。また、管理コンソールおよび管理 REST API でキーを管理する方法にはいくつかの変更があります。詳細は、『サーバー管理ガイド』の「Realm Keys」を参照してください。

シームレスな鍵のローテーションを可能にするには、クライアントアダプターからハードコーディングされたキーを削除する必要があります。レルムキーが指定されていない限り、クライアントアダプターはサーバーからキーを自動的に取得します。クライアントアダプターは、キーがローテーションされると、新しい鍵を自動的に取得します。

2.6.2. クライアントのリダイレクト URI 一致

RH-SSO 7.0 では、クライアントの有効なリダイレクト URI と一致する場合、クエリーパラメーターは無視されます。RH-SSO 7.1 では、クエリーパラメーターは無視されなくなりました。リダイレクト URI にクエリーパラメーターを含める必要がある場合は、クライアントに対して有効なリダイレクト URI で

クエリーパラメーターを指定するか (例: https://hostname/app/login?foo=bar)、ワイルドカードを使用します (例: https://hostname/app/login/*)。フラグメントは、Valid Redirect URI (つまり、https://hostname/app#fragment) でも許可されなくなりました。

2.6.3. Identity Provider への自動リダイレクト

RH-SSO 7.1 では、アイデンティティープロバイダーをデフォルトの認証プロバイダーとして設定することはできません。RH-SSO 7.1 のアイデンティティープロバイダーに自動的にリダイレクトするには、アイデンティティープロバイダーのリダイレクターを設定する必要があります。詳細は、**『Server Administration Guide』**の「Default Identity Provider」 を参照してください。以前デフォルトの認証プロバイダーオプションが設定されたアイデンティティープロバイダーが設定されている場合、この値はサーバーが RH-SSO 7.1 にアップグレードする際にアイデンティティープロバイダーのリダイレクターの値として自動的に使用されます。

2.6.4. 管理 REST API

RH-SSO 7.0 の場合、管理 REST API のページネーションエンドポイントは、maxResults クエリーパラメーターが指定されていない場合に、すべての結果を返します。これにより、一時的に高負荷になる問題が発生し、大量の結果が返されたときにリクエストがタイムアウトする可能性があります (ユーザーなど)。RH-SSO 7.1 では、maxResults の値が指定されていない場合は、最大 100 個の結果が返されます。maxResults を -1 に指定して、すべての結果を返すことができます。

2.6.5. サーバー構成

RH-SSO 7.0 の場合、サーバー設定は keycloak-server.json ファイルおよび standalone/domain.xml または domain.xml ファイルの間で分割されます。RH-SSO 7.1 では、keycloak-server.json ファイルが削除され、すべてのサーバー設定が standalone.xml または domain.xml ファイルで実行されます。RH-SSO 7.1 のアップグレード手順では、サーバー設定を keycloak-server.json ファイルから standalone.xml または domain.xml ファイルに自動的に移行します。

2.6.6. SAML アサーションにおける鍵暗号化アルゴリズム

RH-SSO 7.1 では、SAML アサーションおよびドキュメントのキーは RSA-OAEP 暗号化スキームを使用して暗号化されるようになりました。暗号化されたアサーションを使用するには、サービスプロバイダーがこの暗号化スキームをサポートするようにします。RSA-OAEP をサポートしないサービスプロバイダーがある場合は、システムプロパティー "keycloak.saml.key_trans.rsa_v1.5" を true に設定してサーバーを起動することにより、レガシー RSA-v1.5 暗号化スキームを使用するように RH-SSO を設定できます。これを実行する場合は、よりセキュアな RSA-OAEP 暗号化スキームに戻せるように、できるだけ早くサービスプロバイダーをアップグレードする必要があります。

第3章 RED HAT SINGLE SIGN-ON サーバーのアップグレード

Red Hat Single Sign-On サーバーのアップグレードまたは移行プロセスは、以前のバージョンのソフトウェアによって異なります。

- 7.5.x から 7.6 などのマイナーリリースにアップグレードする場合は、「マイナーアップグレード」の手順に 従います。
- Keycloak 18.0.0 から移行する場合は、「マイナーアップグレード」の手順に従います。
- 7.5.2 から 7.5.3 などのように新しいマイクロリリースにアップグレードする場合は、「マイクロアップグレード」の手順 に従います。

3.1. マイナーアップグレードの実行

3.1.1. アップグレードの準備

アップグレードする前に、アップグレード手順を実行する必要があります。特に、アダプターをアップグレードする前に Red Hat Single Sign-On サーバーをアップグレードするようにしてください。

警告

Red Hat Single Sign-On のマイナーアップグレードでは、すべてのユーザーセッションが失われます。アップグレード後に、すべてのユーザーは再度ログインする必要があります。

手順

1. 以前のインストール (設定、テーマなど) をバックアップします。
2. お使いのリレーショナルデータベースのドキュメントの説明に従い、データベースをバックアップします。
3. Red Hat Single Sign-On サーバーをアップグレードします。
   アップグレード後は、データベースは古いサーバーと互換性がありません。
4. アップグレードを元に戻す必要がある場合は、まず古いインストールを復元してから、バックアップコピーからデータベースを復元します。
5. アダプターをアップグレードします。

3.1.2. Red Hat Single Sign-On サーバーのアップグレード

以下のガイドラインに従って、サーバーが正常にアップグレードされるようにします。

- アップグレードを実稼働以外の環境で最初にテストし、実稼働環境でインストールの問題が発生しないようにします。

- アダプターをアップグレードする前に、Red Hat Single Sign-On サーバーをアップグレードします。また、アダプターをアップグレードする前に、実稼働環境でアップグレードしたサーバーが機能していることを確認します。

警告

このアップグレード手順では、インストールに固有の手動変更により変更が必要になる場合があります。アップグレードに影響する可能性のある手動の変更に関する詳細は、「リリース固有の変更」を参照してください。

インストールに使用した方法に基づいて、サーバーを ZIP ファイル または RPM からアップグレードします。

3.1.2.1. ZIP ファイルからのサーバーのアップグレード

前提条件

- 開かれたトランザクションをすべて処理し、data/tx-object-store/ トランザクションディレクトリーを削除します。

手順

1. 新しいサーバーアーカイブをダウンロードします。
2. ダウンロードしたアーカイブを任意の場所に移動します。
3. アーカイブを展開します。この手順では、最新の Red Hat Single Sign-On リリースのクリーンインスタンスをインストールします。
4. スタンドアロンインストールの場合は、以前のインストールの **RHSSO_HOME/standalone/** ディレクトリーを新しいインストールのディレクトリーにコピーします。
   ドメインのインストールでは、以前のインストールの **RHSSO_HOME/domain/** ディレクトリーを、新しいインストールのディレクトリーにコピーします。

   ドメインのインストールでは、空のディレクトリー **RHSSO_HOME/domain/deployments** を作成します。

   注記: bin ディレクトリーのファイルは、以前のバージョンのファイルで上書きしないでください。変更は手動で行う必要があります。
5. modules ディレクトリーに追加されたカスタムモジュールをコピーします。
6. 「サーバーのアップグレードスクリプトの実行」セクションに進みます。

3.1.2.2. RPM からのサーバーのアップグレード

前提条件

- 開かれたトランザクションをすべて処理し、/var/opt/rh/rh-sso7/lib/keycloak/standalone/data/tx-object-store/ トランザクションディレクトリーを削除します。

手順

1. Red Hat Single Sign-On を含む適切なリポジトリーにサブスクライブします。
Red Hat Enterprise Linux 7 の場合:

```
subscription-manager repos --enable=rh-sso-7.6-for-rhel-7-x86_64-rpms
```

Red Hat Enterprise Linux 8 の場合:

```
subscription-manager repos --enable=rh-sso-7.6-for-rhel-8-x86_64-rpms
```

2. Red Hat Single Sign-On の古い製品リポジトリーを無効にします。

```
subscription-manager repos --disable=rh-sso-7.5-for-rhel-8-x86_64-rpms
```

3. リポジトリーの一覧を確認します。

```
dnf repolist
```

```
Updating Subscription Management repositories.
repo id repo name
rh-sso-7.6-for-rhel-8-x86_64-rpms Single Sign-On 7.6 for RHEL 8 x86_64 (RPMs)
rhel-8-for-x86_64-appstream-rpms Red Hat Enterprise Linux 8 for x86_64 - AppStream (RPMs)
rhel-8-for-x86_64-baseos-rpms Red Hat Enterprise Linux 8 for x86_64 - BaseOS (RPMs)
```

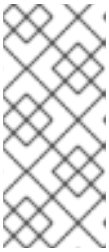
4. 変更した設定ファイルとカスタムモジュールをバックアップします。

5. **dnf upgrade** を使用して、新しい Red Hat Single Sign-On バージョンにアップグレードします。
RPM のアップグレードプロセスでは、変更した設定ファイルは置き換えられません。代わりに、このプロセスにより、新しい Red Hat Single Sign-On バージョンのデフォルト設定に .rpmnew ファイルを作成します。

6. 新しいサブシステムなどの新リリースの新機能を有効にするには、各 .rpmnew ファイルを既存の設定ファイルに手動でマージします。

7. modules ディレクトリーに追加されたカスタムモジュールをコピーします。

8. 「サーバーのアップグレードスクリプトの実行」セクションに進みます。



注記

Red Hat Single Sign-On RPM サーバーディストリビューションが使用している

RHSSO_HOME=/opt/rh/rh-sso7/root/usr/share/keycloak

以下の移行スクリプトを呼び出す場合に使用します。

3.1.3. サーバーのアップグレードスクリプトの実行

以前のインストールに基づいて、お使いの状況に適した適切なアップグレードスクリプトを実行します。

- スタンドアロンモード
- スタンドアロンの高可用性モード
- ドメインモード
- ドメインクラスター化されたモード

3.1.3.1. スタンドアロンモードのアップグレードスクリプトの実行

手順

1. デフォルトの設定ファイルとは異なる設定ファイルを使用している場合は、移行スクリプトを編集して新しいファイル名を指定します。
2. サーバーを停止します。
3. アップグレードスクリプトを実行します。

```
bin/jboss-cli.sh --file=bin/migrate-standalone.cli
```

3.1.3.2. スタンドアロン高可用性モードのアップグレードスクリプトの実行

スタンドアロン高可用性 (HA) モードでは、すべてのインスタンスを同時にアップグレードする必要があります。

手順

1. デフォルトの設定ファイルとは異なる設定ファイルを使用している場合は、移行スクリプトを編集して新しいファイル名を指定します。
2. サーバーを停止します。
3. アップグレードスクリプトを実行します。

```
bin/jboss-cli.sh --file=bin/migrate-standalone-ha.cli
```

3.1.3.3. ドメインモードアップグレードスクリプトの実行

ドメインモードでは、すべてのインスタンスを同時にアップグレードする必要があります。

手順

1. プロファイル名を変更した場合は、アップグレードスクリプトを編集して、スクリプトの最初の方にある変数を変更する必要があります。
2. ドメインスクリプトを編集して、keycloak-server.json ファイルの場所を追加します。
3. サーバーを停止します。

4. ドメインコントローラーでアップグレードスクリプトを実行します。

```
bin/jboss-cli.sh --file=bin/migrate-domain.cli
```

3.1.3.4. ドメインクラスター化されたモードアップグレードスクリプトの実行

ドメインクラスター化されたモードでは、すべてのインスタンスを同時にアップグレードする必要があります。

手順

1. プロファイル名を変更した場合は、アップグレードスクリプトを編集して、スクリプトの最初の方にある変数を変更する必要があります。
2. ドメインクラスター化されたスクリプトを編集して、keycloak-server.json ファイルの場所を追加します。
3. サーバーを停止します。
4. ドメインコントローラーでのみアップグレードスクリプトを実行します。

```
bin/jboss-cli.sh --file=bin/migrate-domain-clustered.cli
```

3.1.4. データベースの移行

Red Hat Single Sign-On は、データベーススキーマを自動的に移行するか、手動で行うこともできます。デフォルトでは、新規インストールを初めて起動すると、データベースが自動的に移行されます。

3.1.4.1. リレーショナルデータベースの自動移行

データベーススキーマの自動アップグレードを有効にするには、migrationStrategy プロパティーの値をデフォルトの connectionsJpa プロバイダーに対して**update**に設定します。

```
<spi name="connectionsJpa">
  <provider name="default" enabled="true">
    <properties>
      ...
      <property name="migrationStrategy" value="update"/>
    </properties>
  </provider>
</spi>
```

または、この CLI コマンドを実行します。

```
/subsystem=keycloak-server/spi=connectionsJpa/provider=default/:map-
put(name=properties,key=migrationStrategy,value=update)
```

この設定でサーバーを起動すると、データベーススキーマが新規バージョンで変更されると、データベースが自動的に移行します。

数百万のレコードを含む大規模テーブルのインデックスを作成すると、簡単に時間がかかる可能性があり、アップグレード時に主要なサービスが中断される可能性があります。このような場合は、自動インデックス作成のしきい値（レコード数）を追加しています。デフォルトでは、このしきい値は **300000**

レコードです。レコードの数がしきい値を上回る場合、インデックスは自動的に作成されず、後で手動で適用できる SQL コマンドを含む、サーバーログに警告メッセージが表示されます。

しきい値を変更するには、**indexCreationThreshold**プロパティ、デフォルトの**connectionsLiquibase**プロバイダーの値を設定します。

```
<spi name="connectionsLiquibase">
  <provider name="default" enabled="true">
    <properties>
      <property name="indexCreationThreshold" value="300000"/>
    </properties>
  </provider>
</spi>
```

または、この CLI コマンドを実行します。

```
/subsystem=keycloak-server/spi=connectionsLiquibase/:add(default-provider=default)
/subsystem=keycloak-server/spi=connectionsLiquibase/provider=default/:add(properties=
{indexCreationThreshold => "300000"},enabled=true)
```

3.1.4.2. 手動によるリレーショナルデータベース移行

データベーススキーマを手動でアップグレードするには、デフォルトの connectionJpa プロバイダーについて migrationStrategy プロパティーの値を**manual**に設定します。

```
<spi name="connectionsJpa">
  <provider name="default" enabled="true">
    <properties>
      ...
      <property name="migrationStrategy" value="manual"/>
    </properties>
  </provider>
</spi>
```

または、この CLI コマンドを実行します。

```
/subsystem=keycloak-server/spi=connectionsJpa/provider=default/:map-
put(name=properties,key=migrationStrategy,value=manual)
```

この設定でサーバーを起動すると、データベースの移行が必要であるかどうかを確認します。必要な変更は、データベースに対して手動で確認および実行できる SQL ファイルに書き込まれます。このファイルをデータベースに適用する方法の詳細は、使用しているリレーショナルデータベースのドキュメントを参照してください。変更がファイルに書き込まれると、サーバーは終了します。

3.1.5. テーマの移行

カスタムテーマを作成している場合は、それらを新しいサーバーに移行する必要があります。組み込みテーマへの変更は、カスタマイズした内容によっては、カスタムテーマに反映する必要がある場合があります。

カスタムテーマを古いサーバーの**themes**ディレクトリーから新しいサーバーの**themes**ディレクトリーにコピーする必要があります。以下の変更を確認し、変更をカスタムテーマに適用する必要があるかどうかを考慮してください。

つまり、以下のようになります。

- 以下に挙げられている変更されたテンプレートのいずれかをカスタマイズした場合は、基本テーマのテンプレートを比較して、適用する必要のある変更があるかどうかを確認する必要があります。
- スタイルをカスタマイズし、Red Hat Single Sign-On を拡張する場合は、スタイルへの変更を確認する必要があります。ベーステーマを拡張する場合は、この手順を省略できます。
- メッセージをカスタマイズする場合は、キーまたは値を変更するか、追加メッセージの追加が必要になる場合があります。

各ステップについて、変更のリストをより詳細に説明しています。

3.1.5.1. Theme changes RH-SSO 7.3

Templates

- Account: account.ftl
- Account: applications.ftl
- Account: resource-detail.ftl (new)
- Account: resources.ftl (new)
- Account: template.ftl
- Account: totp.ftl
- Email-html: email-test.ftl
- Email-html: email-verification-with-code.ftl (new)
- Email-html: email-verification.ftl
- Email-html: event-login_error.ftl
- Email-html: event-removed_totp.ftl
- Email-html: event-update_password.ftl
- Email-html: event-update_totp.ftl
- Email-html: executeActions.ftl
- Email-html: identity-provider-link.ftl
- Email-html: password-reset.ftl
- Email-text: email-verification-with-code.ftl (new)
- Email-text: email-verification.ftl
- Email-text: executeActions.ftl
- Email-text: identity-provider-link.ftl

- Email-text: password-reset.ftl
- Login: cli_splash.ftl (new)
- Login: code.ftl
- Login: error.ftl
- Login: info.ftl
- Login: login-config-totp-text.ftl (new)
- Login: login-config-totp.ftl
- Login: login-idp-link-confirm.ftl
- Login: login-idp-link-email.ftl
- Login: login-oauth-grant.ftl
- Login: login-page-expired.ftl
- Login: login-reset-password.ftl
- Login: login-totp.ftl
- Login: login-update-password.ftl
- Login: login-update-profile.ftl
- Login: login-verify-email-code-text.ftl (new)
- Login: login-verify-email.ftl
- Login: login-x509-info.ftl
- Login: login.ftl
- Login: register.ftl
- Login: template.ftl
- Login: terms.ftl
- Welcome: index.ftl (new)

Messages

- Account: messages_en.properties
- Admin: admin-messages_en.properties
- Email: messages_en.properties
- Login: messages_en.properties

Styles

- Login: login-rhsso.css (new)
- Welcome: welcome-rhsso.css

3.1.5.2. Theme changes RH-SSO 7.2

Templates

- Account: account.ftl
- Account: applications.ftl
- Account: federatedIdentity.ftl
- Account: password.ftl
- Account: sessions.ftl
- Account: template.ftl
- Account: totp.ftl
- Admin: index.ftl
- Email: email-test.ftl (new)
- Email: email-verification.ftl
- Email: event-login_error.ftl
- Email: event-removed_totp.ftl
- Email: event-update_password.ftl
- Email: event-update_totp.ftl
- Email: executeActions.ftl
- Email: identity-provider-link.ftl
- Email: password-reset.ftl
- Login: bypass_kerberos.ftl (removed)
- Login: error.ftl
- Login: info.ftl
- Login: login-config-totp.ftl
- Login: login-idp-link-email.ftl
- Login: login-oauth-grant.ftl
- Login: login-page-expired.ftl (new)
- Login: login-reset-password.ftl

- Login: login-totp.ftl
- Login: login-update-password.ftl
- Login: login-update-profile.ftl
- Login: login-verify-email.ftl
- Login: login-x509-info.ftl (new)
- Login: login.ftl (new)
- Login: register.ftl (new)
- Login: template.ftl (new)
- Login: terms.ftl (new)

Messages

- Account: messages_en.properties
- Admin: admin-messages_en.properties
- Admin: messages_en.properties
- Email: messages_en.properties
- Login: messages_en.properties

Styles

- Account: account.css
- Login: login.css

3.1.5.3. Theme changes RH-SSO 7.1

Templates

- Account: account.ftl
- Account: federatedIdentity.ftl
- Account: totp.ftl
- Login: info.ftl
- Login: login-config-totp.ftl
- Login: login-reset-password.ftl
- Login: login.ftl

Messages

- Account: editAccountHtmlTtile renamed to editAccountHtmlTitle

- Account: role_uma_authorization added
- Login: loginTotpStep1 value changed
- Login: invalidPasswordGenericMessage added
- Login: invlidRequesterMessage renamed to invalidRequesterMessage
- Login: clientDisabledMessage added

Styles

- Account: account.css
- Login: login.css

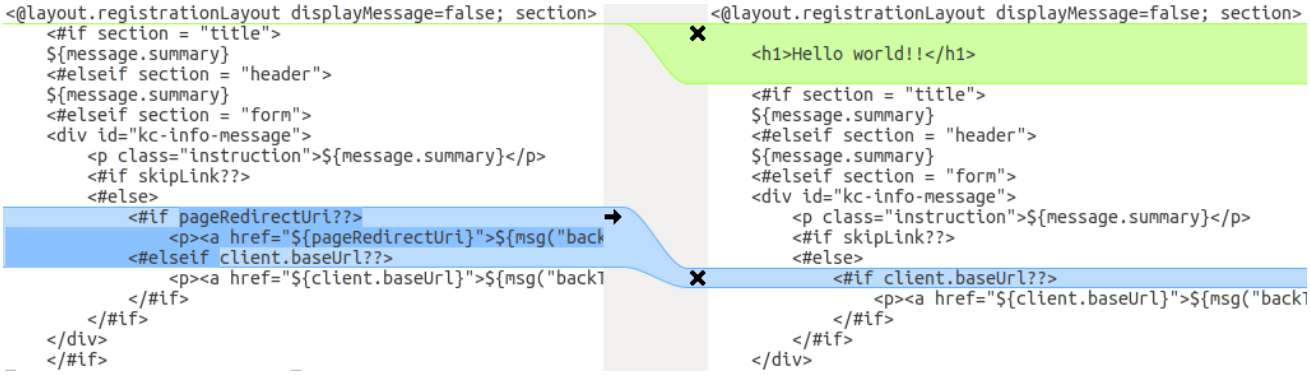
3.1.5.4. テンプレートの移行

テンプレートのいずれかをカスタマイズする場合は、テンプレートに加えた変更を慎重に確認して、カスタマイズされたテンプレートにこれらの変更を適用する必要があるかどうかを判断します。カスタマイズされたテンプレートに同じ変更を適用する必要がある可能性が高いです。一覧表示されたテンプレートをカスタマイズしていない場合は、このセクションを飛ばすことができます。

ベストプラクティスとして、diff ツールを使用してテンプレートを比較し、カスタマイズされたテンプレートに実行する必要のある変更を確認できます。マイナーな変更のみを行った場合は、更新されたテンプレートをカスタマイズされたテンプレートと比較することが簡単になります。ただし、多くの変更を加えた場合は、新しいテンプレートをカスタマイズされた古いテンプレートと比較することが容易になるかもしれません。これにより、どのような変更が必要になるかが分かります。

次のスクリーンショットは、ログインテーマの info.ftl テンプレートとサンプルのカスタムテーマを比較しています。

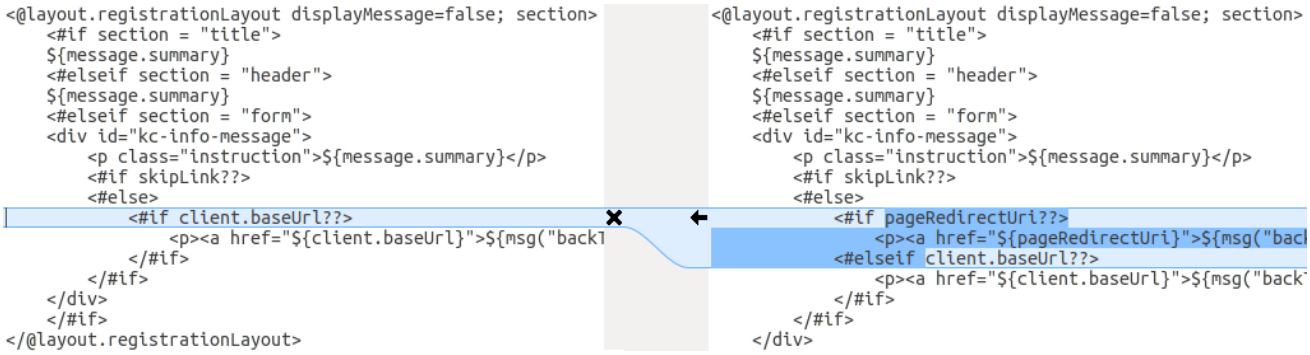
ログインテーマテンプレートの更新バージョンとサンプルのカスタムログインテーマテンプレートの比較



この比較から、最初の変更 (**Hello world!!**) がカスタマイズされ、2 番目の変更 (**if pageRedirectUri**) がベーステーマに変更されていることを簡単に特定することができます。2 つ目の変更をカスタムテンプレートにコピーすることにより、カスタマイズされたテンプレートが正常に更新されました。

別の方法としては、以下のスクリーンショットでは、古いインストールの info.ftl テンプレートと、新しいインストールから更新された info.ftl テンプレートを比較します。

ログインテーマテンプレートと、更新されたログインテーマテンプレートの比較



この比較から、ベーステンプレートで変更されたものを簡単に特定できます。次に、変更したテンプレートに対して同じ変更を加える必要があります。このアプローチは最初のアプローチほど単純ではないため、最初のアプローチが実行可能でない場合にのみこのアプローチを使用してください。

3.1.5.5. メッセージの移行

別の言語のサポートを追加する場合は、上記のすべての変更を適用する必要があります。別の言語のサポートを追加していなかった場合は、何も変更する必要がない可能性があります。自身のテーマで影響を受けるメッセージを変更した場合に限り、変更を加える必要があります。

追加された値については、ベーステーマのメッセージ値を確認し、メッセージをカスタマイズする必要があるかどうかを判断します。

名前が変更された鍵の場合は、カスタムテーマのキーの名前を変更します。

変更された値については、ベーステーマの値を確認して、カスタムテーマに変更を加えなければならないかどうかを判断します。

3.1.5.6. スタイルの移行

keycloak または rh-sso テーマからスタイルを継承している場合は、組み込みテーマからスタイルに加えられた変更を反映するようにカスタムスタイルの更新が必要になる場合があります。

ベストプラクティスは、diff ツールを使用して、古いサーバーインストールと新しいサーバーインストールとの間でスタイルシートへの変更を比較することです。

たとえば、diff コマンドを使用します。

```
$ diff RHSSO_HOME_OLD/themes/keycloak/login/resources/css/login.css \
RHSSO_HOME_NEW/themes/keycloak/login/resources/css/login.css
```

変更を確認し、それらがカスタムスタイルに影響するかどうかを判断します。

3.2. マイクロアップグレードの実行

3.2.1. インストールのパッチ適用

RH-SSO の ZIP インストールのパッチは、Red Hat カスタマーポータル からダウンロードできます。

管理対象ドメイン環境の複数の RH-SSO ホストの場合、各ホストは RH-SSO ドメインコントローラーからパッチを当てることができます。

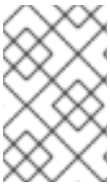
パッチ適用の他に、パッチの適用をロールバックすることもできます。

3.2.1.1. ZIP インストールパッチに関する重要事項

- モジュールを更新するパッチを適用すると、起動時に使用される新しいパッチが適用された JAR は **RHSSO_HOME/modules/system/layers/base/.overlays/PATCH_ID/MODULE** に保存されます。パッチが適用されていない元のファイルは **RHSSO_HOME/modules/system/layers/base/MODULE** に残りますが、これらの JAR は起動時に **使用されません**。

- RH-SSO 7 の累積パッチリリースのサイズを大幅に減らすため、累積パッチを部分的にロールバックすることはできません。適用済みのパッチはパッチ全体のみをロールバックできます。たとえば、CP03 を RH-SSO 7.0.0 に適用する場合は、CP01 または CP02 にロールバックすることはできません。各累積パッチリリースにロールバックする機能が必要な場合は、各累積パッチをリリースされた順序で個別に適用する必要があります。

3.2.1.2. パッチの適用



注記

RPM 方式を使用してインストールされた RH-SSO サーバーは、これらの手順を使用して更新することはできません。代わりに、「パッチを適用する RPM の手順」を参照してください。

管理 CLI または 管理コンソール のいずれかを使用して、ダウンロードしたパッチを RH-SSO サーバーに適用できます。

手順

1. Red Hat カスタマーポータル (https://access.redhat.com/downloads/) からパッチファイルをダウンロードします。
2. 管理 CLI から、パッチファイルへの適切なパスを含む以下のコマンドを使用してパッチを適用します。

```
patch apply /path/to/downloaded-patch.zip
```

注記

管理対象ドメインの別の RH-SSO ホストにパッチを適用するには、**--host=** 引数を使用して RH-SSO ホスト名を指定できます。以下に例を示します。

```
patch apply /path/to/downloaded-patch.zip --host=my-host
```

パッチの適用時に競合が存在する場合は、パッチツールによって警告が表示されます。競合が存在する場合は、利用可能な引数に **patch --help** を入力し、競合の解決方法を指定する引数を使用してコマンドを再実行します。

3. RH-SSO サーバーを再起動して、パッチを適用します。

```
shutdown --restart=true
```

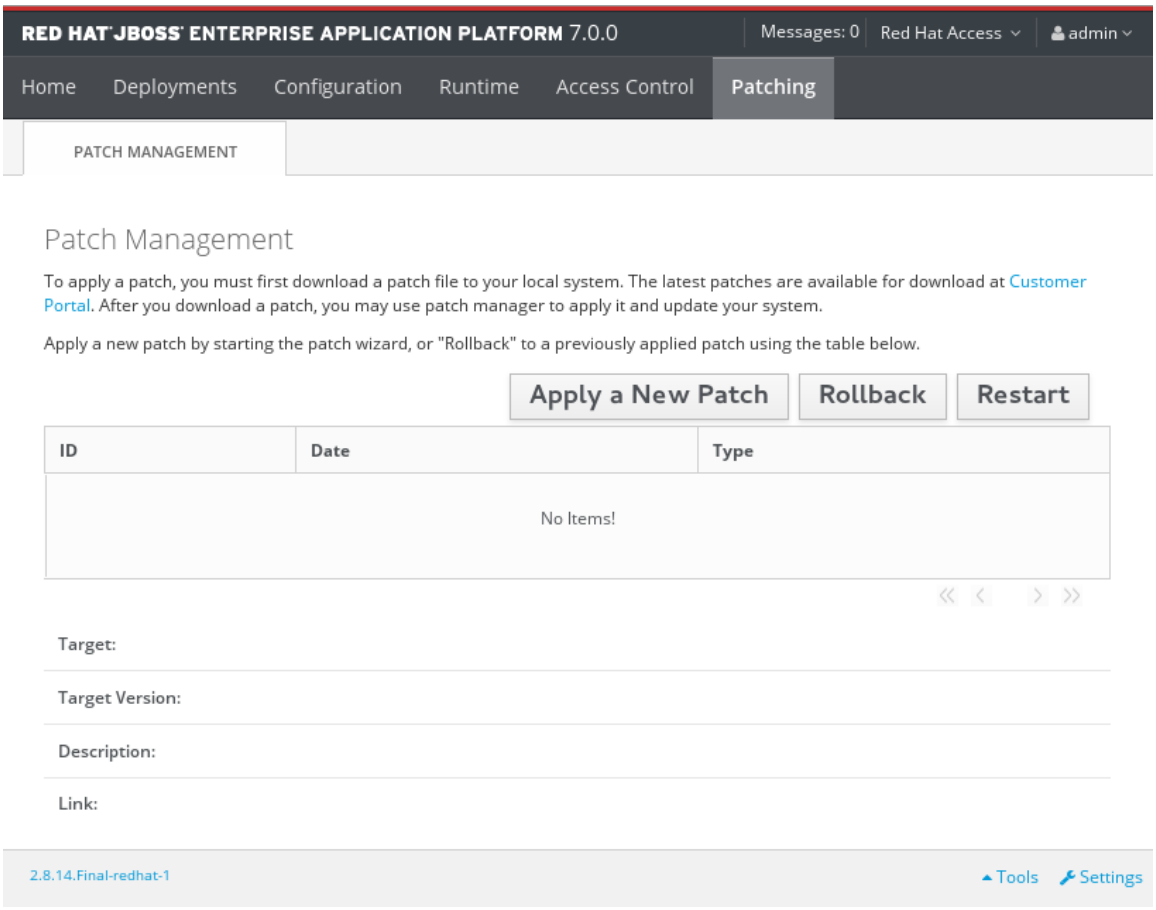
手順

1. Red Hat カスタマーポータル (https://access.redhat.com/downloads/) からパッチファイルをダウンロードします。

2. 管理コンソール を開き、**Patch Management** ビューに移動します。

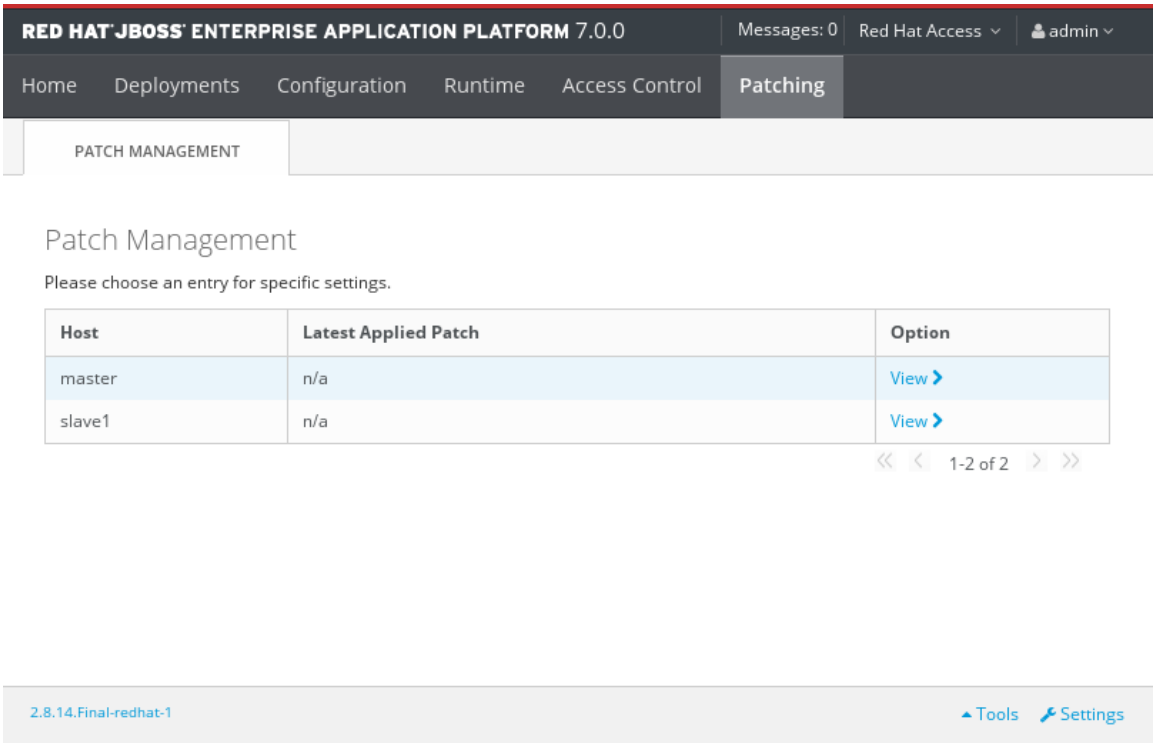
   a. スタンドアロンサーバーの場合は、**Patching** タブをクリックします。

   スタンドアロンサーバーのパッチ管理画面



   b. 管理対象ドメインのサーバーの場合は **Patching** タブをクリックし、表からパッチを適用するホストを選択して **View** をクリックします。

   管理対象ドメインのパッチ管理画面

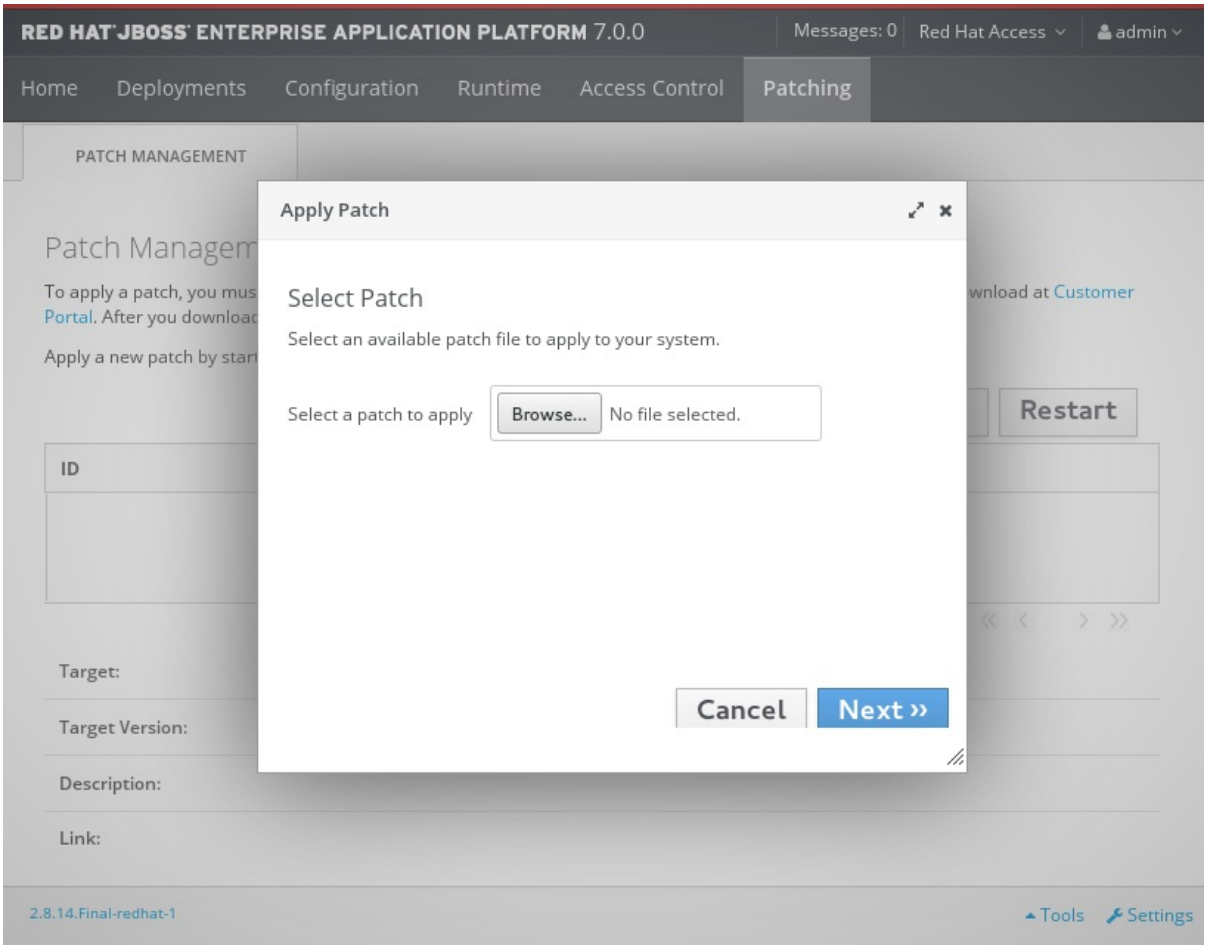


3. **Apply a New Patch** をクリックします。

    a. 管理対象ドメインホストにパッチを当てる場合は、次の画面で、ホストのサーバーをシャットダウンするかどうかを選択し、**Next** をクリックします。

4. **Browse** ボタンをクリックして適用するダウンロードしたパッチを選択し、**Next** をクリックします。

パッチ画面の適用



..パッチの適用の試行に競合が存在する場合は、警告が表示されます。**View error details** をクリックして、競合の詳細を確認します。競合が存在する場合は、操作をキャンセルするか、**Override all conflicts** を選択し、**Next** をクリックします。競合を上書きすると、パッチのコンテンツがユーザーの変更を上書きします。

5. パッチの適用が正常に完了したら、今すぐ RH-SSO を再起動するかどうかを選択し、パッチを適用するかどうかを選択し、**Finish** をクリックします。

3.2.1.3. パッチのロールバック

管理 CLI または 管理コンソール を使用して、以前適用した RH-SSO パッチをロールバックできます。

重要

パッチ管理システムを使用したパッチのロールバックは、一般的なアンインストール機能として意図されていません。これは、望ましくない影響を及ぼしたパッチの適用直後にのみ使用することを目的としています。

要件

- 以前に適用されたパッチ。

警告

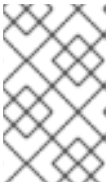
いずれかの手順を行場合は、**Reset Configuration** オプションの値を指定する際に注意して行ってください。

TRUE に設定されていると、パッチのロールバックプロセスによって RH-SSO サーバー設定ファイルもパッチ前の状態にロールバックされます。パッチの適用後に RH-SSO サーバー構成ファイルに加えられた変更はすべて失われます。

FALSE に設定すると、サーバー設定ファイルはロールバックされません。この状況では、パッチによって名前空間などの構成が変更されている可能性があるため、ロールバック後にサーバーが起動しない可能性があります。名前空間は無効になり、手動で修正する必要があります。

手順

1. 管理 CLI から **patch history** コマンドを使用して、ロールバックするパッチの ID を見つけます。



注記

管理対象ドメインを使用している場合は、この手順のコマンドに **--host=HOSTNAME** 引数を追加して、RH-SSO ホストを指定する必要があります。

2. 直前の手順の適切なパッチ ID でパッチをロールバックします。

```
patch rollback --patch-id=PATCH_ID --reset-configuration=TRUE
```

パッチのロールバック時に競合が存在する場合は、パッチツールによって警告が表示されます。競合が存在する場合は、利用可能な引数に **patch --help** を入力し、競合の解決方法を指定する引数を使用してコマンドを再実行します。

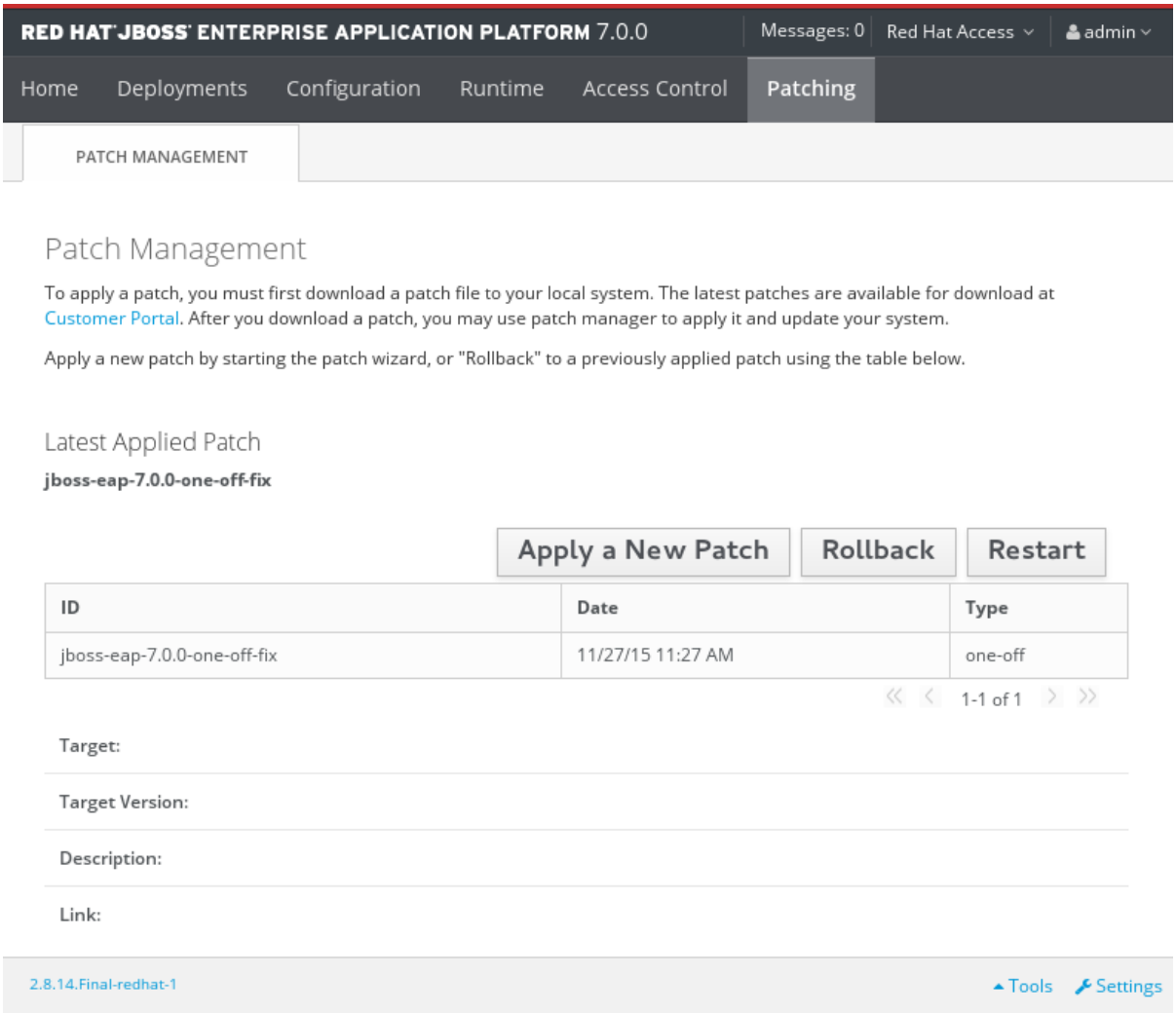
3. パッチのロールバックに RH-SSO サーバーを再起動します。

```
shutdown --restart=true
```

手順

1. 管理コンソールを開き、**Patch Management** ビューに移動します。
   a. スタンドアロンサーバーの場合は、**Patching** タブをクリックします。
   b. 管理対象ドメインのサーバーの場合は **Patching** タブをクリックし、表からパッチを適用するホストを選択して **View** をクリックします。
2. 表に一覧表示されているものからロールバックするパッチを選択し、**Rollback** をクリックします。

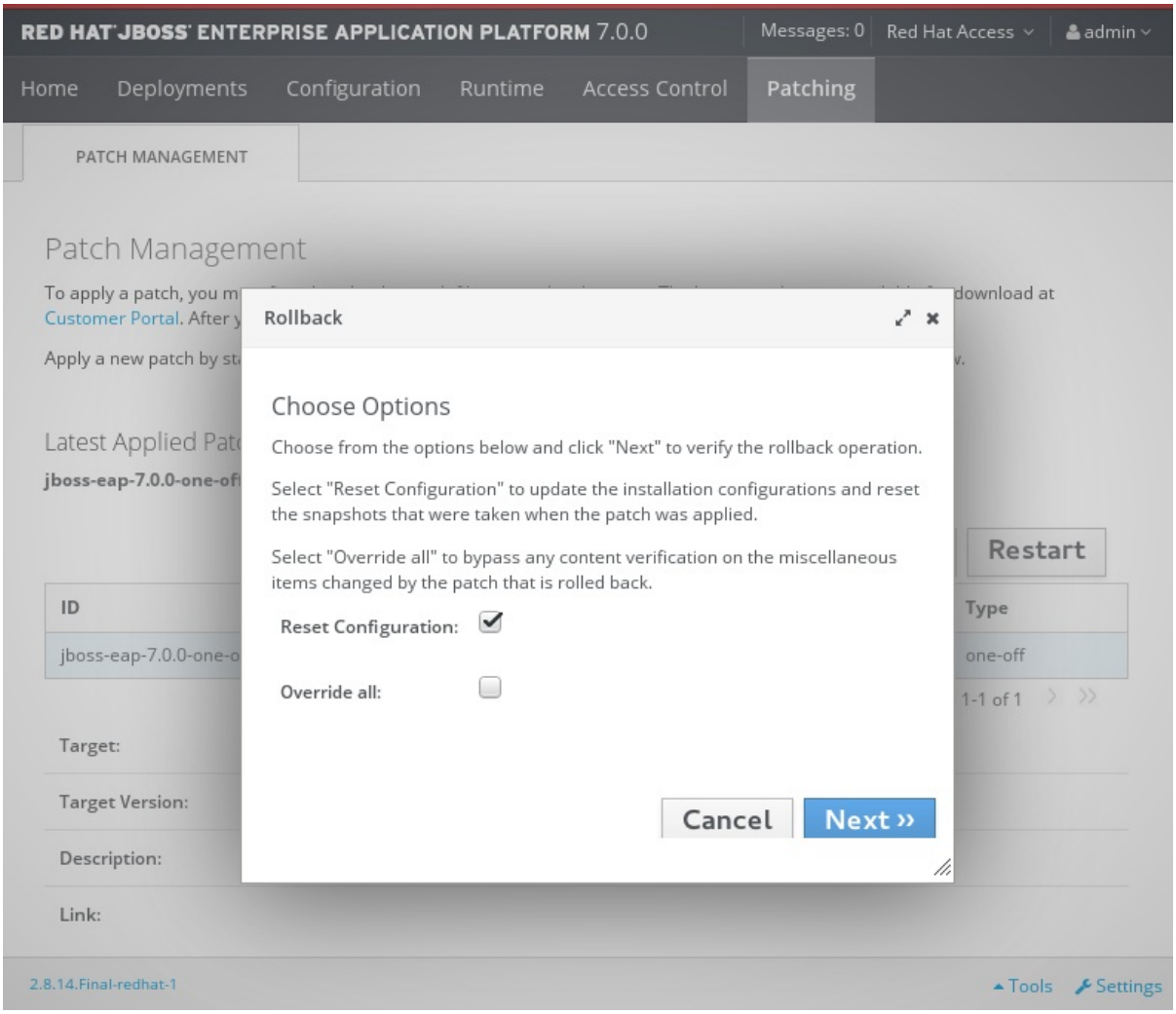
最新のパッチ履歴画面



..管理対象ドメインホストでパッチをロールバックする場合は、次の画面で、ホストのサーバーをシャットダウンするかどうかを選択し、**Next** をクリックします。

3. ロールバックプロセスのオプションを選択して、**Next** をクリックします。

パッチのロールバックオプション



4. ロールバックするオプションとパッチを確認してから **Next** をクリックします。

    a. パッチのロールバック時に競合が発生し、**Override all** オプションが選択されなかった場合は、警告が表示されます。**View error details** をクリックして、競合の詳細を確認します。競合が存在する場合は、操作をキャンセルするか、**Choose Options** をクリックし、**Override all** チェックボックスを選択して操作を再試行します。競合を上書きすると、ロールバック操作でユーザーの変更がオーバーライドされます。

5. パッチが正常にロールバックされたら、変更を有効にするために RH-SSO サーバーを再起動するかどうかを選択し、**Finish** をクリックします。

3.2.1.4. パッチ履歴の消去

パッチが RH-SSO サーバーに適用されると、ロールバック操作で使用するためにパッチの内容と履歴が保持されます。複数の累積パッチが適用されている場合、パッチ履歴が使用するディスク領域はかなりの量になる場合があります。

以下の管理 CLI コマンドを使用すると、現在使用されていない古いパッチをすべて削除することができます。このコマンドを使用する場合は、GA リリースとともに最新の累積パッチのみが保持されます。これは、これまでに複数の累積パッチが適用されている場合にのみ領域を解放するのに役立ちます。

```
/core-service=patching:ageout-history
```

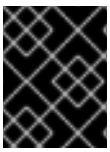
重要

パッチ履歴を消去すると、これあまでに適用されたパッチをロールバックできなくなります。

3.2.2. RPM インストールへのパッチ適用

要件

- ベースのオペレーティングシステムが最新の状態で、標準の Red Hat Enterprise Linux リポジトリーにサブスクライブし、更新を取得できるようにしてください。
- 更新に関連する RH-SSO リポジトリーにサブスクライブしていることを確認してください。
- 設定ファイル、デプロイメント、およびユーザーデータをすべてバックアップする



重要

管理対象ドメインでは、RH-SSO ドメインコントローラーを最初に更新する必要があります。

サブスクライブしているリポジトリーから RPM 経由で RH-SSO パッチをインストールするには、以下のコマンドを使用して Red Hat Enterprise Linux システムを更新します。

```
yum update
```

第4章 RED HAT SINGLE SIGN-ON アダプターのアップグレード

まず Red Hat Single Sign-On サーバーをアップグレードしてから、アダプターをアップグレードすることが重要です。アダプターの以前のバージョンは、新しいバージョンの Red Hat Single Sign-On サーバーと連携しますが、Red Hat Single Sign-On サーバーの以前のバージョンは、新しいバージョンのアダプターでは機能しない可能性があります。

4.1. 古いアダプターとの互換性

上記のように、以前のリリースバージョンのアダプターを使用する新しいリリースバージョンの Red Hat Single Sign-On サーバーのサポートを試みます。ただし、Red Hat Single Sign-On サーバー側への修正の追加が必要になる場合があります。これにより、古いバージョンのアダプターとの互換性が損なわれる場合があります。たとえば、OpenID Connect 仕様の新しいアスペクトを実装する場合に、古いクライアントアダプターバージョンは認識されませんでした。

このような場合は、互換性モードが追加されました。OpenId Connect クライアントでは、クライアント詳細が含まれるページに、Red Hat Single Sign-On 管理コンソールの **OpenID Connect Compatibility Modes** という名前のセクションがあります。ここでは、古いクライアントアダプターとの互換性を維持するために、Red Hat Single Sign-On サーバーの新しい機能を無効にすることができます。詳細については、個々のスイッチのツールチップを参照してください。

4.2. EAP アダプターのアップグレード

手順

ダウンロードしたアーカイブを使用してアダプターを最初にインストールした場合、JBoss EAP アダプターをアップグレードするには、以下の手順を実行します。

1. 新しいアダプターアーカイブをダウンロードします。
2. **EAP_HOME/modules/system/add-ons/keycloak/** ディレクトリーを削除して、以前のアダプターモジュールを削除します。
3. ダウンロードしたアーカイブを **EAP_HOME** に展開します。

手順

最初に RPM を使用してアダプターをインストールした場合、アダプタをアップグレードするには、次の手順を実行します。これらの手順は、マイナーアップグレードとマイクロアップグレードのどちらを実行しているかによって異なります。

1. マイナーアップグレードの場合は、Yum を使用して、現在インストールされているアダプターをアンインストールしてから、Yum を使用して、新しいバージョンのアダプターをインストールします。
2. マイクロアップグレードでは、Yum を使用してアダプターをアップグレードします。これは、マイクロアップグレードの唯一の手順です。

```
yum update
```

4.3. JAVASCRIPT アダプターのアップグレード

Web アプリケーションにコピーされた JavaScript アダプターをアップグレードするには、以下の手順を実行します。

手順

1. 新しいアダプターアーカイブをダウンロードします。
2. アプリケーションの keycloak.js ファイルは、ダウンロードしたアーカイブの keycloak.js ファイルを上書きします。

4.4. NODE.JS アダプターのアップグレード

Web アプリケーションにコピーされた Node.js アダプターをアップグレードするには、以下の手順を実行します。

手順

1. 新しいアダプターアーカイブをダウンロードします。
2. 既存の Node.js アダプターディレクトリーを削除します。
3. 更新されたファイルをその場所に展開します。
4. アプリケーションの package.json で keycloak-connect の依存関係を変更します。