Red Hat JBoss Enterprise Application Platform 6.4

サーバーセキュリティーの設定方法

サーバーセキュリティーの設定方法

Last Updated: 2023-02-21

Red Hat JBoss Enterprise Application Platform 6.4 サーバーセキュリティーの設定方法

サーバーセキュリティーの設定方法

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

Copyright © 2022 | You need to change the HOLDER entity in the en-US/How_To_Configure_Server_Security.ent file |.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at
http://creativecommons.org/licenses/by-sa/3.0/
. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, the Red Hat logo, JBoss, OpenShift, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux ® is the registered trademark of Linus Torvalds in the United States and other countries.

Java ® is a registered trademark of Oracle and/or its affiliates.

XFS ® is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL ® is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js ® is an official trademark of Joyent. Red Hat is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack ® Word Mark and OpenStack logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

概要

本書の目的は、Red Hat JBoss Enterprise Application Platform (JBoss EAP) のセキュリティーを保護するための実用的なガイドを提供することです。具体的には、JBoss EAP 6 のすべての管理インターフェイスをセキュアにする方法を説明します。本ガイドを読む前に、Red Hat JBoss Enterprise Application Platform のセキュリティーアーキテクチャーを読み、JBoss EAP によるセキュリティーの処理方法を理解してください。また、本書では JBoss EAP CLI インターフェースを使用して設定の変更を行います。スタンドアロンの JBossEAP6 インスタンスと JBossEAP6 ドメ

インの両方で CLI を使用する方法の詳細については、管理および設定ガイドの管理 CLI セクションを参照してください。本書を完読することで、JBoss EAP 6 をセキュアにする方法についてしっかりと理解することができます。

目次

第1章 セキュリティーの概要

JBoss EAP 6 セキュリティーの基本と一般的なセキュリティーの概念は、Red Hat JBoss Enterprise Application Platform セキュリティーアーキテクチャーガイド で説明されています。このガイドを読む前に、認証、承認、セキュリティーレルム、暗号化、SSL/TLS に関するセキュリティーアーキテクチャーガイドで説明されている基本情報を理解することが重要です。

第2章 サーバーおよびインターフェースの保護

2.1. ブロックの構築

2.1.1. インターフェースおよびソケットバインディング

JBoss EAP 6 は、ホストのインターフェイス (inet-address、nic など) とポートを使用して、Web アプリケーションと管理インターフェイスの両方の通信を行います。これらのインターフェイスとポートは、JBoss EAP 6 設定ファイル (**standalone.xml**、**domain.xml**、**host.xml** など) の **インターフェイス** と **socket-binding-groups** 設定を介して定義および設定されます。**インターフェイス** と **ソケット** バインディンググループを定義および設定する方法の詳細については、Red Hat JBoss Enterprise Application Platform 6 管理および設定ガイド の インターフェイス と ソケットバインディンググループ セクションを参照してください。

例: インターフェース

```
<interfaces>
  <interface name="management">
    <inet-address value="${jboss.bind.address.management:127.0.0.1}"/>
  </interface>
  <interface name="public">
    <inet-address value="${jboss.bind.address:127.0.0.1}"/>
  </interface>
  <interface name="unsecure">
    <inet-address value="${jboss.bind.address.unsecure:127.0.0.1}"/>
  </interface>
</interfaces>
```

例: ソケットバインディンググループ

```
<socket-binding-group name="standard-sockets" default-interface="public" port-
offset="${jboss.socket.binding.port-offset:0}">
  <socket-binding name="management-native" interface="management"
port="${jboss.management.native.port:9999}"/>
  <socket-binding name="management-http" interface="management"
port="${jboss.management.http.port:9990}"/>
  <socket-binding name="management-https" interface="management"
port="${jboss.management.https.port:9443}"/>
  <socket-binding name="ajp" port="8009"/>
  <socket-binding name="http" port="8080"/>
  <socket-binding name="https" port="8443"/>
  <socket-binding name="remoting" port="4447"/>
  <socket-binding name="txn-recovery-environment" port="4712"/>
  <socket-binding name="txn-status-manager" port="4713"/>
  <outbound-socket-binding name="mail-smtp">
    <remote-destination host="localhost" port="25"/>
  </outbound-socket-binding>
</socket-binding-group>
```

2.1.2. セキュリティーレルム

JBoss EAP はセキュリティーレルムを使用して、ローカル、LDAP、プロパティーなどの認証および承認メカニズムを定義します。セキュリティーレルムの背景情報は、Red Hat JBoss Enterprise

Application Platform『セキュリティーアーキテクチャー』ガイドの「**セキュリティレルム**」を参照してください。

例: セキュリティーレルム

```
<security-realms>
  <security-realm name="ManagementRealm">
    <authentication>
      <local default-user="$local" skip-group-loading="true"/>
      <properties path="mgmt-users.properties" relative-to="jboss.server.config.dir"/>
    </authentication>
    <authorization map-groups-to-roles="false">
      <properties path="mgmt-groups.properties" relative-to="jboss.server.config.dir"/>
    </authorization>
  </security-realm>
  <security-realm name="ApplicationRealm">
    <authentication>
      <local default-user="$local" allowed-users="*" skip-group-loading="true"/>
      <properties path="application-users.properties" relative-to="jboss.server.config.dir"/>
    </authentication>
    <authorization>
      <properties path="application-roles.properties" relative-to="jboss.server.config.dir"/>
    </authorization>
  </security-realm>
</security-realms>
```

注記

JBoss EAP 6 では、既存のセキュリティーレルムを更新するだけでなく、新しいセキュリティーレルムを作成して使用することもできます。新しいセキュリティーレルムは、管理コンソールを介して作成することも、管理 CLI から次のコマンドを呼び出すこともできます。**/core-service=management/security-realm=NEW-REALM-NAME:add ()**

2.1.3. 管理インターフェイスを保護するためのセキュリティーレルムとソケットバインディングの使用

JBoss EAP 6 は、2 つの異なる管理インターフェイスを定義します。

- http-interface
- native-interface

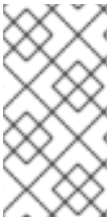
これらのインターフェイスは、JBossEAP6 設定の **<management-interfaces>** セクションで定義されています。

```
<management-interfaces>
  <native-interface security-realm="ManagementRealm">
    <socket-binding native="management-native"/>
  </native-interface>
  <http-interface security-realm="ManagementRealm">
    <socket-binding http="management-http"/>
  </http-interface>
</management-interfaces>
```

各インターフェイスが **セキュリティーレルム** と **ソケットバインディング** を指定していることに注意してください。指定されたセキュリティーレルムとソケットバインディングの設定を更新すると、管理インターフェイスをさまざまな方法で保護できます。セキュリティーレルムとソケットバインディングを介してこれらの各インターフェイスを保護できることに加えて、これらのインターフェイスの両方を完全に無効にすることもできます。また、これらのインターフェイスのユーザーは、さまざまなロールとアクセス権を持つように設定できます。このガイドには、セキュリティー監査、安全なパスワード、JBoss EAP 6 内の他のサブシステムと重複する JMX など、JBossEAP6 の保護に関連するトピックもいくつかあります。

2.2. 管理インターフェースのセキュア化の方法

ここでは、JBoss EAP 管理インターフェースと関連サブシステムのセキュア化に関連するさまざまな操作を実行する方法を説明します。



注記

以下の CLI コマンドは、JBossEAP6 のスタンドアロンインスタンスを想定して実行されました。JBoss EAP 6 ドメインでの CLI の使用の詳細については、Red Hat JBoss Enterprise Application Platform 管理およびガイド**の管理 CLI** セクションを参照してください。

2.2.1. Red Hat JBoss Enterprise Application Platform で使用されるネットワークとポートの設定

ホストの設定に応じて、JBoss EAP はさまざまなネットワークインターフェースおよびポートを使用するように設定できます。これにより、JBoss EAP はさまざまなホスト、ネットワーク、およびファイアウォール要件で使用できます。JBoss EAP 6 で使用されるネットワークとポート、およびそれらの設定の設定方法の詳細については、Red Hat JBoss Enterprise Application Platform 6 管理および設定ガイドの **ネットワークとポートの設定** セクションを参照してください。

2.2.2. HTTPS 用に管理コンソールを設定

HTTPS 経由でのみ通信するように JBoss EAP 管理コンソールを設定すると、セキュリティーが強化されます。クライアント (Web ブラウザー) と管理コンソール間のネットワークトラフィックはすべて暗号化されるため、仲介者攻撃などのセキュリティー攻撃のリスクが軽減されます。

この手順では、JBossEAP6 スタンドアロンインスタンスまたはドメインとの暗号化されていない通信が無効になります。この手順は、スタンドアロン設定とドメインモード設定の両方に適用されます。ドメインモードの場合、管理 CLI コマンドのプレフィックスに、例えば以下のようにホスト名を付けます。例: **/host=master**。

HTTPS 用に管理コンソールを設定するには、次の手順が必要です。

1. キーストアを作成して、管理インターフェースをセキュア化
2. 管理コンソールが HTTPS にバインドされていることの確認
3. オプション: カスタムソケットバインディンググループ
4. 新しいセキュリティーレルムの作成
5. 新しいセキュリティーレルムを使用するように管理インターフェイスを設定
6. キーストアを使用するように管理コンソールを設定

7. Red Hat JBoss Enterprise Application Platform インスタンスの再起動

2.2.2.1. 1.キーストアを作成して、管理インターフェースをセキュア化

注記

このキーストアは、管理コンソールが JCEKS 形式のキーストアと互換性がないため、JKS 形式である必要があります。

以下を実行してキーストアを生成します。パラメーターの例の値 (例: alias、keypass、keystore、storepass 、dname) を環境に適切な値に置き換えます。

注記

パラメーターの **validity** は、キーが有効な日数を指定します。730 の値は 2 年と等しくなります。

keytool コマンドを使用して端末からキーストアを生成する

```
keytool -genkeypair -alias appserver -storetype jks -keyalg RSA -keysize 2048 -keypass password1 -keystore EAP_HOME/standalone/configuration/identity.jks -storepass password1 -dname "CN=appserver,OU=Sales,O=Systems Inc,L=Raleigh,ST=NC,C=US" -validity 730 -v
```

2.2.2.2. 2.管理コンソールが HTTPS にバインドされていることの確認

スタンドアロンモード

management-https 設定を追加し、**management-http** 設定を削除して、管理コンソールがインターフェイスの HTTPS にバインドしていることを確認します。

次の CLI コマンドを使用して、管理コンソールを HTTPS にバインドします。

```
/core-service=management/management-interface=http-interface:write-attribute( \
name=secure-socket-binding, value=management-https)
```

```
/core-service=management/management-interface= \
http-interface:undefine-attribute(name=socket-binding)
```

注記

この時点で、JBoss EAP 6 ログに次のエラーメッセージが表示される場合があります 。**JBAS015103:HTTP インターフェイスにセキュアポートが指定されていますが、レルムに SSL 設定がありません。**SSL 設定がまだ完了していないため、これは予想されることです。

ドメインモード

secure-portを追加し、ポート設定を削除して、**management-interface** セクション内のソケット要素を変更します。

次のコマンドを使用して、管理コンソールを HTTPS にバインドします。

```
/host=master/core-service=management/management-interface= \
http-interface:write-attribute(name=secure-port,value=9443)
```

```
/host=master/core-service=management/management-interface= \
http-interface:undefine-attribute(name=port)
```

注記

この時点で、JBoss EAP 6 ログに次のエラーメッセージが表示される場合があります。**JBAS015103:HTTP インターフェイスにセキュアポートが指定されていますが、レルムに SSL 設定がありません。**SSL 設定がまだ完了していないため、これは予想されることです。

2.2.2.3. 3.オプション: カスタムソケットバインディンググループ

カスタムソケットバインディンググループが使用されている場合は、management-https バインディングが定義されていることを確認します (デフォルトで存在し、ポート 9443 にバインドされています)。マスター設定ファイル (**standalone.xml** など) を次のように編集します。

XML の例

```
<socket-binding-group name="standard-sockets" default-interface="public"
    port-offset="${jboss.socket.binding.port-offset:0}">
  <socket-binding name="management-native" interface="management"
    port="${jboss.management.native.port:9999}"/>
  <socket-binding name="management-http" interface="management"
    port="${jboss.management.http.port:9990}"/>
  <socket-binding name="management-https" interface="management"
    port="${jboss.management.https.port:9443}"/>
...
```

2.2.2.4. 4.新しいセキュリティーレルムの作成

この例では、HTTPS の **ManagementRealmHTTPS** を使用する新しいセキュリティーレリムは、EAP_HOME/standalone/configuration/ ディレクトリーに位置する **https-mgmt-users.properties** というプロパティーファイルを使用します。ユーザー名とパスワードは後でファイルに追加されますが、今のところ、**https-mgmt-users.properties** という名前の空のファイルを作成し、その場所に保存するだけです。次の例は、端末からの **touch** コマンドの使用を示していますが、他のメカニズム (テキストエディターの使用など) も使用できます。

ターミナルから touch コマンドを使用して空のファイルを作成する例

```
touch EAP_HOME/standalone/configuration/https-mgmt-users.properties
```

次に、次の CLI コマンドを入力して、ManagementRealmHTTPS という名前の新しいセキュリティーレルムを作成します。

```
/core-service=management/security-realm=ManagementRealmHTTPS/:add
```

```
/core-service=management/security-realm=ManagementRealmHTTPS/authentication= \
properties/:add(path=https-mgmt-users.properties, \
relative-to=jboss.server.config.dir)
```

新しいプロパティーファイルとレルムが作成されたので、レルムで使用するには、ユーザーをそのプロパティーファイルに追加する必要があります。これは、**EAP_HOME/bin/** ディレクトリーにある **add-user** スクリプトを介して実行されます。**add-user** を実行するときは、プロパティーファイルとレルムの両方をそれぞれ **-up** オプションと **-r** オプションを使用して指定する必要があります。そこから、add-user スクリプトは、https-mgmt-users.properties ファイルに保存するためのユーザー名とパスワードの情報を対話的に要求します。

```
EAP_HOME/bin/add-user.sh -up EAP_HOME/standalone/configuration/https-mgmt-users.properties -r ManagementRealmHTTPS
...
Enter the details of the new user to add.
Using realm 'ManagementRealmHTTPS' as specified on the command line.
...
Username : httpUser
Password requirements are listed below. To modify these restrictions edit the add-user.properties configuration file.
 - The password must not be one of the following restricted values {root, admin, administrator}
 - The password must contain at least 8 characters, 1 alphabetic character(s), 1 digit(s), 1 non-alphanumeric symbol(s)
 - The password must be different from the username
...
Password :
Re-enter Password :
About to add user 'httpUser' for realm 'ManagementRealmHTTPS'
...
Is this correct yes/no? yes
..
Added user 'httpUser' to file 'EAP_HOME/configuration/https-mgmt-users.properties'
...
Is this new user going to be used for one AS process to connect to another AS process?
e.g. for a slave host controller connecting to the master or for a Remoting connection for server to server EJB calls.
yes/no? no
```

重要

プロパティーファイルを使用してユーザー名とパスワードを保存するようにセキュリティレルムを設定する場合、各レリムは別のレルムと共有されていない個別のプロパティーを使用することが推奨されます。

2.2.2.5. 5.新しいセキュリティーレルムを使用するように管理インターフェイスを設定

新しいセキュリティーレルムを使用するための CLI コマンド

```
/core-service=management/management-interface=http-interface/:write-attribute( \
name=security-realm,value=ManagementRealmHTTPS)
```

2.2.2.6. 6.キーストアを使用するように管理コンソールを設定

以下の CLI コマンドを使用して、キーストアを使用するように管理コンソールを設定します。パラメーターファイル、パスワード、エイリアスの場合、それらの値を 最初のステップ からコピーする必要があります。

キーストアをセキュリティーレルムに追加するための CLI コマンド

```
/core-service=management/security-realm=ManagementRealmHTTPS/server-identity= \
ssl:add(keystore-path=identity.jks, \
keystore-relative-to=jboss.server.config.dir, \
keystore-password=password1, alias=appserver)
```

```
reload
```

注記

キーストアのパスワードを更新するには、以下の CLI コマンドを使用します。

```
/core-service=management/security-realm=ManagementRealmHTTPS/server-identity=ssl:write-attribute(name=keystore-password,value=newpassword)
```

2.2.2.7. 7.Red Hat JBoss Enterprise Application Platform インスタンスの再起動

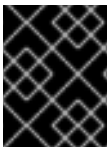
サーバーを再起動すると、ログには、開始されたサービスの数を示すテキストの直前に、次のものが含まれている必要があります。これで、管理コンソールはポート 9443 をリッスンするようになります。これにより、手順が成功したことを確認できました。

```
14:53:14,720 INFO  [org.jboss.as] (Controller Boot Thread) JBAS015962: Http management interface listening on https://127.0.0.1:9443/management
14:53:14,721 INFO  [org.jboss.as] (Controller Boot Thread) JBAS015952: Admin console listening on https://127.0.0.1:9443
```

2.2.3. 個別の HTTP および HTTPS 管理インターフェイスのセットアップ

管理インターフェイスは、HTTP と HTTPS の接続を別々のインターフェイスでリッスンすることができます。このシナリオの 1 つは、外部ネットワークで暗号化されたトラフィックをリッスンし、内部ネットワークで暗号化されていないトラフィックを使用することです。

secure-interface 属性は、**interface** 属性で指定されたものとは異なるインターフェイスを使用する必要がある場合に、HTTPS 管理通信用のホストのソケットを開く必要があるネットワークインターフェイスを指定します。指定されていない場合は、**interface** 属性で指定されたインターフェイスが使用されます。



重要

secure-port 属性が設定されていない場合、secure **-interface** 属性は効果がありません。

サーバーが同じインターフェースの HTTP および HTTPS トラフィックをリッスンする場合には、HTTP リスナーによって受信される HTTPS リクエストは自動的に HTTPS ポートにリダイレクトされる点に注意してください。HTTP および HTTPS トラフィックに個別のインターフェースが使用される場合、HTTP リスナーが HTTPS 要求を受信する際にリダイレクトは実行されません。

次に、HTTP トラフィックとは異なるインターフェイスで HTTPS トラフィックをリッスンするように **secure-interface** 属性を設定する **EAP_HOME/domain/configuration/host.xml** 設定の例を示します。

XML の例

```
<host name="master" xmlns="urn:jboss:domain:3.0">
```

```
    <management>
        <security-realms>
            <security-realm name="ManagementRealm">
                <authentication>
                    <local default-user="$local" />
                    <properties path="mgmt-users.properties"
                      relative-to="jboss.domain.config.dir"/>
                </authentication>
            </security-realm>
        </security-realms>
        <management-interfaces>
            <native-interface security-realm="ManagementRealm">
                <socket interface="management"
                  port="${jboss.management.native.port:9999}"/>
            </native-interface>
            <http-interface security-realm="ManagementRealm">
                <socket interface="management"
                  port="${jboss.management.http.port:9990}"
                  secure-port="${jboss.management.https.port:9943}"
                  secure-interface="secure-management"/>
            </http-interface>
        </management-interfaces>
    </management>
    <domain-controller>
        <local/>
        <!-- Alternative remote domain controller configuration with a host and port -->
        <!-- <remote host="${jboss.domain.master.address}" port="${jboss.domain.master.port:9999}"
security-realm="ManagementRealm"/> -->
    </domain-controller>
    <interfaces>
        <interface name="management">
            <inet-address value="${jboss.bind.address.management:127.0.0.1}"/>
        </interface>
        <interface name="secure-management">
            <inet-address value="${jboss.bind.address:10.10.64.1}"/>
        </interface>
    </interfaces>
</host>
```

2.2.4. HTTP インターフェイスの無効化

管理対象ドメインや特定の実稼働環境展開などの特定のシナリオでは、管理者は HTTP を介した管理インターフェイスへのアクセスを無効にするか、HTTP 管理インターフェイスを完全に削除したい場合があります。

以下を使用して、HTTP インターフェイスの現在の設定を読み取ることができます。

現在の HTTP インターフェイス設定を読み取るための CLI コマンド

```
/core-service=management/management-interface=http-interface/:read-resource( \
recursive=true, proxies=false, include-runtime=false, include-defaults=true)
```

結果

```
{
```

```
    "outcome" => "success",
    "result" => {
        "console-enabled" => true,
        "interface" => "management",
        "port" => expression "${jboss.management.http.port:9990}",
        "secure-port" => undefined,
        "security-realm" => "ManagementRealm"
    }
}
```

HTTP インターフェイスを完全に削除するには:

HTTP インターフェイスを完全に削除するための CLI コマンド

```
/core-service=management/management-interface=http-interface/:remove
```

HTTP インターフェイスを削除したら、それを再作成することで再度有効にすることができます。

HTTP インターフェイスを再作成するための CLI コマンド:

```
/core-service=management/management-interface=http-interface:add( \
console-enabled=true, interface=management, \
port="${jboss.management.http.port:9990}", \
security-realm=ManagementRealm)
```

管理コンソールのみを無効にする

JBoss Operations Network などの他のクライアントは、JBoss EAP の管理に HTTP インターフェースを使用して動作します。これらのサービスの使用を継続するには、Web ベースの管理コンソール自体を無効にしてください。これは、**console-enabled** 属性を **false** に設定すると実行できます。

Web ベースの管理コンソールを無効にするための CLI コマンド

```
/core-service=management/management-interface=http-interface/:write-attribute( \
name=console-enabled,value=false)
```

2.2.5. 管理インターフェイス用の双方向 SSL/TLS の設定

クライアント認証 とも呼ばれる双方向 SSL/TLS 認証は、SSL/TLS 証明書を使用してクライアントとサーバーの両方を認証します。これは、クライアントとサーバーの両方にそれぞれ証明書があるという点で 、HTTPS 用の管理コンソールの設定 セクションとは異なります。そのため、サーバーの伝えるアイデンティティーだけでなく、クライアントの伝えるアイデンティティーも信頼できます。

本セクションでは、以下の規則を使用します。

HOST1

JBoss サーバーのホスト名。例: jboss.redhat.com

HOST2

クライアントに適した名前。例: myclientこれは必ずしも実際のホスト名ではないことに注意してください。

CA_HOST1

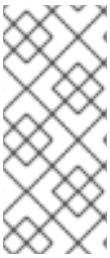
HOST1 証明書に使用する DN (識別名) です。例: cn=jboss,dc=redhat,dc=com

CA_HOST2

HOST2 証明書に使用する DN (識別名) です。例: cn=myclient,dc=redhat,dc=com

管理インターフェイスの双方向 SSL/TLS を設定するには、次の手順が必要です。

1. ストアを生成します。
2. 証明書をエクスポートします。
3. もう一方のトラストストアに証明書をインポートします。
4. CertificateRealm を定義します。
5. ネイティブインターフェイスのセキュリティーレルムを新しい証明書レルムに変更します。
6. CLI の SSL 設定を追加します



前提条件

キーストアおよびトラストストアパスワードの保存にパスワード vault が使用された場合 (推奨)、パスワード vault はすでに作成されている必要がある。パスワード vault の詳細は、Red Hat JBoss Enterprise Application Platform 6 Security Architecture guide の formation on the password vault, please see the Password Vault セクションと **Password Vault System** セクションを参照。

1.ストアを生成します。

```
keytool -genkeypair -alias HOST1_alias -keyalg RSA -keysize 1024 -validity 365 -keystore HOST1.keystore.jks -dname "CA_HOST1" -keypass secret -storepass secret
```

```
keytool -genkeypair -alias HOST2_alias -keyalg RSA -keysize 1024 -validity 365 -keystore HOST2.keystore.jks -dname "CA_HOST2" -keypass secret -storepass secret
```

2.証明書をエクスポートします。

```
keytool -exportcert  -keystore HOST1.keystore.jks -alias HOST1_alias -keypass secret -storepass secret -file HOST1.cer
```

```
keytool -exportcert  -keystore HOST2.keystore.jks -alias HOST2_alias -keypass secret -storepass secret -file HOST2.cer
```

3.もう一方のトラストストアにインポートします。

```
keytool -importcert -keystore HOST1.truststore.jks -storepass secret -alias HOST2_alias -trustcacerts -file HOST2.cer
```

```
keytool -importcert -keystore HOST2.truststore.jks -storepass secret -alias HOST1_alias -trustcacerts -file HOST1.cer
```

4.CertificateRealm を定義します。

インストール (**host.xml または standalone.xml**) の設定で CertificateRealm を定義し、その先となるインターフェースを指定します。これは、以下のコマンドで実行できます。

```
/core-service=management/security-realm=CertificateRealm:add()
```

```
/core-service=management/security-realm=CertificateRealm/server-identity= \
ssl:add(keystore-path=/path/to/HOST1.keystore.jks, keystore-password=secret, \
alias=HOST1_alias)
```

```
/core-service=management/security-realm=CertificateRealm/authentication= \
truststore:add(keystore-path=/path/to/HOST1.truststore.jks, \
keystore-password=secret)
```

5.ネイティブインターフェイスのセキュリティーレルムを新しい証明書レルムに変更します。

```
/core-service=management/management-interface= \
native-interface:write-attribute(name=security-realm,value=CertificateRealm)
```

6.CLI の SSL 設定を追加します。

EAP_HOME/bin/jboss-cli.xml を設定ファイルとして使用する CLI の SSL 設定を追加します。トラストストアとキーストアのパスワードは、次の 2 つの方法のいずれかで保存できます。

- パスワードボールト (推奨)
- プレインテキスト

6A。キーストアとトラストストアのパスワードをパスワードボールトに保存するには:

EAP_HOME/bin/jboss-cli.xml を編集し、SSL/TLS 設定を追加します (変数に適切な値を使用)。また、ボールト設定を追加し、各値を使用されているボールトの値に置き換えます。

jboss-cli.xmlXML の例

```
<ssl>
  <vault>
    <vault-option name="KEYSTORE_URL" value="path-to/vault/vault.keystore"/>
    <vault-option name="KEYSTORE_PASSWORD" value="MASK-5WNXs8oEbrs"/>
    <vault-option name="KEYSTORE_ALIAS" value="vault"/>
    <vault-option name="SALT" value="12345678"/>
    <vault-option name="ITERATION_COUNT" value="50"/>
    <vault-option name="ENC_FILE_DIR" value="EAP_HOME/vault/"/>
  </vault>
  <alias>HOST2_alias</alias>
  <key-store>/path/to/HOST2.keystore.jks</key-store>
  <key-store-password>VAULT::VB::cli_pass::1</key-store-password>
  <key-password>VAULT::VB::cli_pass::1</key-password>
  <trust-store>/path/to/HOST2.truststore.jks</trust-store>
  <trust-store-password>VAULT::VB::cli_pass::1</trust-store-password>
  <modify-trust-store>true</modify-trust-store>
</ssl>
```

6B。キーストアとトラストストアのパスワードをプレーンテキストで保存するには:

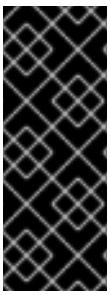
EAP_HOME/bin/jboss-cli.xml を編集し、SSL/TLS 設定を追加します (変数に適切な値を使用)。

jboss-cli.xmlXML の例

```
<ssl>
  <alias>HOST2_alias</alias>
  <key-store>/path/to/HOST2.keystore.jks</key-store>
  <key-store-password>secret</key-store-password>
  <trust-store>/path/to/HOST2.truststore.jks</trust-store>
  <trust-store-password>secret</trust-store-password>
  <modify-trust-store>true</modify-trust-store>
</ssl>
```

2.2.6. SSL/TLS コネクターの設定

管理インターフェイスで HTTPS および双方向 SSL/TLS をサポートすることに加えて、JBoss EAP 6 では、セキュリティードメインで使用するために SSL/TLS コネクターをセットアップすることもできます。



重要

前提条件として、SSL/TLS 暗号化キーと証明書を作成し、アクセス可能なディレクトリーに配置する必要があります。さらに、関連情報 (たとえば、キーストアのエイリアスとパスワード、必要な暗号スイートなど) にもアクセスできる必要があります。SSL/TLS キーおよび証明書の生成例は、「方向 SSL/TLS の設定 」セクションにある最初の手順を参照してください。SSL/TLS コネクター (暗号スイートを含む) の詳細については、SSL コネクターリファレンスセクション を参照してください。

Web サブシステムに HTTPS コネクターを追加します

https スキーム、https ソケットバインディング (デフォルトは 8443) を使用し、安全に設定されている **HTTPS** という名前の安全なコネクターを作成します。

```
/subsystem=web/connector=HTTPS/:add(socket-binding=https,scheme=https, \
protocol=HTTP/1.1,secure=true)
```

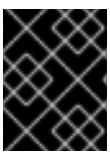
例の値を正しい値に置き換えて、SSL 証明書を設定します。この例では、キーストアがサーバー設定ディレクトリーにコピーされていることを前提としています。サーバー設定ディレクトリーは、**スタンドアロンの JBossEAP6 インスタンスの場合は EAP_HOME/Standalone/configuration/** です。

```
/subsystem=web/connector=HTTPS/ssl=configuration:add( \
name=https,certificate-key-file="${jboss.server.config.dir}/keystore.jks", \
password=SECRET, key-alias=KEY_ALIAS, cipher-suite=CIPHERS)
```

プロトコルを **TLSv1** に設定します。

```
/subsystem=web/connector=HTTPS/ssl=configuration/:write-attribute( \
name=protocol,value=TLSv1)
```

アプリケーションをデプロイします。



重要

これらの変更を有効にするには、JBossEAP6 インスタンスを再起動する必要があります。

2.2.7. SSL コネクターリファレンス

JBoss Web コネクターには、次の SSL 設定属性が含まれる場合があります。

2.2.7.1. name

SSL コネクターの表示名。属性名は読み取り専用です。

2.2.7.2. verify-client

HTTP/HTTPS コネクターまたはネイティブ APR コネクターが使用されるかによって、verify-client の可能な値は異なります。

HTTP/HTTPS コネクター

可能な値は、**true**、**false**、または **want です**。**true** に設定すると、接続を受け入れる前にクライアントからの有効な証明書チェーンが必要になります。SSL スタックがクライアント証明書を要求する **か** どうかを指定しますが、提示されない場合は失敗しません。クライアントが CLIENT-CERT 認証を使用するセキュリティー制約によって保護されたリソースを要求しない限り、証明書チェーンを必要としないようにするには、**false** (デフォルト) に設定します。

CLI の例

```
/subsystem=web/connector=HTTPS/ssl=configuration/:write-attribute(name=verify-client,value=want)
```

ネイティブ APR コネクター

可能な値は、**optional**、**require**、**optionalNoCA**、および **none** (または **none** と同じ効果を持つ他の文字列) です。これらの値は、認証がオプションであるか、必須であるか、認証局なしでオプションであるか、またはまったく不要であるかを決定します。デフォルトは none です。これは、クライアントが証明書を送信する機会がないことを意味します。

CLI の例

```
/subsystem=web/connector=APR/ssl=configuration/:write-attribute( \
name=verify-client,value=require)
```

2.2.7.3. verify-depth

クライアントが有効な証明を持たないと判断するまでにチェックされる中間証明書発行者の最大数。デフォルト値は 10 です。

CLI の例

```
/subsystem=web/connector=HTTPS/ssl=configuration/:write-attribute( \
name=verify-depth,value=10)
```

2.2.7.4. certificate-key-file

署名されたサーバー証明書が保存されているキーストアファイルの完全なファイルパスとファイル名。JSSE 暗号化では、この証明書ファイルが 1 つだけになりますが、OpenSSL は複数のファイルを使用します。デフォルト値は、JBossEAP6 を実行しているユーザーのホームディレクトリーにある.keystore ファイルです。keystoreType がファイルを使用しない場合は、パラメーターを空の文字列に設定します。

CLI の例

```
/subsystem=web/connector=HTTPS/ssl=configuration/:write-attribute( \
name=certificate-key-file, \
value=../domain/configuration/server.keystore)
```

2.2.7.5. certificate-file

OpenSSL 暗号化を使用する場合は、このパラメーターの値を、サーバー証明書を含むファイルに対するパスに設定します。

CLI の例

```
/subsystem=web/connector=HTTPS/ssl=configuration/:write-attribute( \
name=certificate-file,value=server.crt)
```

2.2.7.6. password

トラストストアとキーストアの両方のパスワード。次の例では、PASSWORD を実際のパスワードに置き換えます。

CLI の例

```
/subsystem=web/connector=HTTPS/ssl=configuration/:write-attribute( \
name=password,value=PASSWORD)
```

2.2.7.7. protocol

使用する SSL プロトコルのバージョン。サポートされる値は、基盤となる SSL 実装 (JSSE または OpenSSL) によって異なります。JavaSSE ドキュメント を参照してください。

プロトコルのコンマ区切りの組み合わせも指定できます。例: TLSv1, TLSv1.1,TLSv1.2。

警告

Red Hat は、影響を受けるすべてのパッケージで TLSv1.1 または TLSv1.2 を優先して、SSL を明示的に無効にすることを推奨しています。

CLI の例

```
/subsystem=web/connector=HTTPS/ssl=configuration/:write-attribute( \
name=protocol,value=ALL)
```

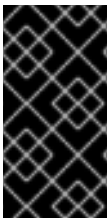
```
/subsystem=web/connector=HTTPS/ssl=configuration/:write-attribute( \
name=protocol,value="TLSv1, TLSv1.1,TLSv1.2")
```

2.2.7.8. cipher-suite

許可される暗号化暗号のリスト。JSSE 構文については、コンマ区切りのリストが必要です。OpenSSL 構文については、コロン区切のリストが必要です。1 つのみの構文を使うようにしてください。

デフォルトは **HIGH:!aNULL:!eNULL:!EXPORT:!DES:!RC4:!MD5** です。

この例では、2 つの暗号だけが一覧表示されますが、実際の例はより多くの暗号が使用されます。



重要

強度の低い暗号を使用すると、セキュリティが重大なリスクにさらされることになります。暗号スイートに関する NIST の推奨事項については、http://www.nist.gov/manuscript-publication-search.cfm?pub_id=915295 を参照してください。

使用可能な OpenSSL 暗号のリストについては、https://www.openssl.org/docs/manmaster/apps/ciphers.html#CIPHER-STRINGS を参照してください。以下はサポート対象外であることに注意してください。

- **@SECLEVEL**
- **SUITEB128**
- **SUITEB128ONLY**
- **SUITEB192**

標準の JSSE 暗号のリストについては、http://docs.oracle.com/javase/7/docs/technotes/guides/security/StandardNames.html#Cipher を参照してください。

CLI の例

```
/subsystem=web/connector=HTTPS/ssl=configuration/:write-attribute( \
name=cipher-suite, \
value="TLS_RSA_WITH_AES_128_CBC_SHA,TLS_RSA_WITH_AES_256_CBC_SHA")
```

2.2.7.9. key-alias

キーストアのサーバー証明書に使用されるエイリアス。次の例では、KEY_ALIAS を証明書のエイリアスに置き換えます。

CLI の例

```
/subsystem=web/connector=HTTPS/ssl=configuration/:write-attribute( \
name=key-alias,value=KEY_ALIAS)
```

2.2.7.10. truststore-type

トラストストアのタイプ。PKCS12 や Java の標準 JKS など、さまざまなタイプのトラストストアを利用できます。

CLI の例

```
/subsystem=web/connector=HTTPS/ssl=configuration/:write-attribute( \
name=truststore-type,value=jks)
```

2.2.7.11. keystore-type

キーストアのタイプ。PKCS12 や Java の標準 JKS など、さまざまなタイプのキーストアを利用できます。

CLI の例

```
/subsystem=web/connector=HTTPS/ssl=configuration/:write-attribute( \
name=keystore-type,value=jks)
```

2.2.7.12. ca-certificate-file

CA 証明書を含むファイル。これは、JSSE の場合は truststoreFile であり、キーストアと同じパスワードを使用します。ca-certificate-file ファイルは、クライアント証明書を検証するために使用されます。

CLI の例

```
/subsystem=web/connector=HTTPS/ssl=configuration/:write-attribute( \
name=ca-certificate-file,value=ca.crt)
```

2.2.7.13. ca-certificate-password

ca-certificate-file の証明書パスワード。次の例では、MASKED_PASSWORD をマスクされたパスワードに置き換えます。

CLI の例

```
/subsystem=web/connector=HTTPS/ssl=configuration/:write-attribute( \
name=ca-certificate-password, value=MASKED_PASSWORD)
```

2.2.7.14. ca-revocation-url

失効リストを含むファイルまたは URL。これは、JSSE の場合は crlFile、SSL の場合は SSLCARevocationFile を指します。

CLI の例

```
/subsystem=web/connector=HTTPS/ssl=configuration/:write-attribute( \
name=ca-revocation-url,value=ca.crl)
```

2.2.7.15. session-cache-size

SSLSession キャッシュのサイズ。この属性は、JSSE コネクターにのみ適用されます。デフォルトは 0 で、無制限のキャッシュサイズを指定します。

CLI の例

```
/subsystem=web/connector=HTTPS/ssl=configuration/:write-attribute( \
name=session-cache-size,value=100)
```

2.2.7.16. session-timeout

キャッシュされた SSLSession が期限切れになるまでの秒数。この属性は、JSSE コネクターにのみ適用されます。デフォルトは 86400 秒、つまり 24 時間です。

CLI の例

```
/subsystem=web/connector=HTTPS/ssl=configuration/:write-attribute( \
name=session-timeout,value=43200)
```

2.2.8. JMX へのリモートアクセスの無効化

JMX サブシステムへのリモートアクセスにより、JDK とアプリケーションの管理操作をリモートでトリガーできます。JBoss EAP 6 で JMX へのリモートアクセスを無効にするには、次の 2 つのいずれかを実行する必要があります。

1. JMX サブシステムのリモートコネクターを削除します
2. JMX サブシステム全体を削除します

リモーティングコネクターの削除

```
/subsystem=jmx/remoting-connector=jmx/:remove
```

JMX サブシステム全体を削除する

```
/subsystem=jmx/:remove
```

JMX の詳細は、Red Hat JBoss Enterprise Application Platform『セキュリティーアーキテクチャー』の「JMX」セクションを参照してください。

2.2.9. 管理インターフェースのセキュア化での JAAS の使用

JAAS は、JBoss EAP がセキュリティーを管理するために使用する宣言型セキュリティー API です。JAAS と宣言型セキュリティーの詳細は、『Red Hat JBoss Enterprise Application Platform Security Architecture』ガイドの「宣言型セキュリティーと JAAS」を参照します。

注記

JBoss EAP 6 インスタンスが **ADMIN_ONLY** モードで実行されるように設定されていると、JAAS を使用した管理インターフェースのセキュア化はサポートされません。**ADMIN_ONLY** モードについて詳しくは、管理設定ガイド の **サーバー実行時に渡すスイッチと引数のリファレンス** を参照してください。

JAAS を使用して管理インターフェースに認証するには、以下の手順を実行する必要があります。

1. セキュリティードメインを作成します。
2. JAAS 認証でセキュリティーレルムを作成します。

3. 新しいセキュリティーレルムを使用するように管理インターフェイスを更新します
4. **オプション:** グループメンバーシップを割り当てます。

1.セキュリティードメインの作成

この例では、セキュリティードメインが UserRoles ログインモジュールで作成されますが、その他のログインモジュールも使用できます。

```
/subsystem=security/security-domain=UsersLMDomain:add( \
cache-type=default)
```

```
/subsystem=security/security-domain=UsersLMDomain/authentication=classic:add
```

```
/subsystem=security/security-domain=UsersLMDomain/authentication= \
classic/login-module=UsersRoles:add(code=UsersRoles, flag=required, \
module-options=[("usersProperties"=>"users.properties"), \
("rolesProperties"=>"roles.properties")])
```

2.JAAS 認証でセキュリティーレルムを作成します。

JAAS 認証でセキュリティーレルムを作成します。

```
/core-service=management/security-realm=SecurityDomainAuthnRealm:add
```

```
/core-service=management/security-realm= \
SecurityDomainAuthnRealm/authentication=jaas:add(name=UsersLMDomain)
```

3.新しいセキュリティーレルムを使用するように管理インターフェイスを更新します

新しいレルムを使用するように **http-interface** を更新するには:

```
/core-service=management/management-interface=http-interface/:write-attribute( \
name=security-realm,value=SecurityDomainAuthnRealm)
```

新しいレルムを使用するように **ネイティブインターフェイス** を更新するには、次の手順に従います。

```
/core-service=management/management-interface=native-interface/:write-attribute( \
name=security-realm,value=SecurityDomainAuthnRealm)
```

4.オプション: グループメンバーシップを割り当てます。

属性 assign-groups は、セキュリティーレルム内のグループ割り当てに、セキュリティードメインからロードされたユーザーメンバーシップ情報を使用するかどうかを決定します。true に設定すると、グループ割り当ては、Role-Based Access Control (RBAC) に使用されます。

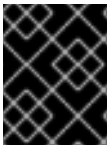
割り当てグループ属性を設定するには、次のようにします。

```
/core-service=management/security-realm=SecurityDomainAuthnRealm/authentication= \
jaas:write-attribute(name=assign-groups,value=true)
```

2.2.10. サイレント認証

JBoss EAP 6 のデフォルトインストールには、ローカル管理 CLI ユーザーのサイレント認証メソッドが含まれています。これにより、ローカルユーザーは、ユーザー名またはパスワード認証なしで管理 CLI にアクセスできます。この機能は利便性のためと、管理 CLI スクリプトを実行しているローカルユーザー認証を不要にするために有効になっています。通常、ユーザーがローカル設定にアクセスできれば、独自のユーザー詳細を追加することもできるため便利な機能と見なされます。

ローカルユーザーのサイレント認証の利便性は、セキュリティー管理が長い場合に無効にすることができます。これは、設定ファイルの security-realm セクション内の local 要素を削除することで実現できます。これは、スタンドアロンインスタンスとドメインの両方に適用されます。



重要

local 要素の削除は、JBoss EAP インスタンスの影響と、その設定を完全に理解している場合にのみ行う必要があります。

レルムからサイレント認証を削除するには:

```
/core-service=management/security-realm=REALM_NAME/authentication=local:remove
```

2.3. セキュリティー監査

セキュリティー監査とは、セキュリティーサブシステムまたは管理インターフェイス内で発生したイベントに応答して、ログへの書き込みなどのイベントをトリガーすることを指します。監査メカニズムは、セキュリティードメインまたは管理インターフェイスの一部として設定されます。

監査は、プロバイダーモジュールを使用します。含まれるプロバイダーモジュールとカスタム実装の両方を使用できます。

2.3.1. 管理インターフェイスのセキュリティー監査を設定する

管理インターフェイスの監査の設定の詳細については、Red Hat JBoss Enterprise Application Platform 6 管理および設定ガイド の 管理インターフェイス監査ログセクション を参照してください。

2.3.2. セキュリティードメインのセキュリティー監査を設定する

セキュリティードメインのセキュリティー監査設定を設定するには、管理コンソールから以下の手順を実行する必要があります。

1. セキュリティードメインの詳細ビューを開く
2. 監査サブシステム設定に移動します。
3. プロバイダーモジュールを追加します。
4. モジュールが機能していることを確認します
5. モジュールオプションを追加、編集、または削除します (任意)。<remark>Bugzilla のバグの内容を反映済み</remark>

1.セキュリティードメインの詳細ビューを開く

- 画面上部の **設定** をクリックします。

- 管理対象ドメインで、左上の **プロファイル** 選択ボックスから変更するプロファイルを選択します。
- **セキュリティー** メニューを展開し、**セキュリティードメイン** を選択します。
- 編集するセキュリティードメインの **表示** をクリックします。

2.監査サブシステム設定に移動します。

画面上部の監査タブを選択します。

設定領域は、プロバイダーモジュールと詳細の 2 つの領域に分かれています。プロバイダーモジュールは、設定の基本単位です。セキュリティードメインには、属性とオプションを含めることができる複数のプロバイダーモジュールを含めることができます。

3.プロバイダーモジュールを追加します。

追加 をクリックして、**コード** セクションにプロバイダーモジュールのクラス名を入力します。

4.モジュールが機能していることを確認します

監査モジュールの目標は、security サブシステムでイベントをモニターする手段を提供することです。このモニタリングは、ログファイル、電子メール通知、またはその他の測定可能な監査メカニズムに書き込む方法で実行できます。

たとえば、JBoss EAP 6 には、デフォルトで **org.jboss.security.audit.providers.LogAuditProvider** モジュールが含まれています。上記の手順に従って有効にした場合、この監査モジュールは、**EAP_HOME** ディレクトリー内のログサブフォルダーにある **audit.log** ファイルにセキュリティー通知を書き込みます。

上記の手順が **org.jboss.security.audit.providers.LogAuditProvider** のコンテキストで機能したかどうかを確認するには、通知をトリガーする可能性のあるアクションを実行してから、監査ログファイルを確認します。

モジュールオプションを追加、編集、または削除します (任意)。<remark>Bugzilla のバグの内容を反映済み</remark>

モジュールにオプションを追加するには、**モジュール** リストでそのエントリーをクリックし、ページの **詳細** セクションで **モジュールオプション** タブを選択します。**追加** をクリックして、オプションのキーと値を指定します。

すでに存在するオプションを編集するには、**削除** をクリックして削除し、**追加** をクリックして正しいオプションで再度追加します。

第3章 サーバーのユーザーと管理インターフェースのセキュア化

JBoss EAP 6 のさまざまなインターフェイスを保護する方法を理解することに加えて、それらのインターフェイスにアクセスするユーザーを保護する方法を理解することも重要です。

3.1. ユーザー認証

3.1.1. デフォルトのユーザー設定

JBoss EAP のすべての管理インターフェースはデフォルトでセキュアになります。また、ユーザーはローカルインターフェースとリモートインターフェースという 2 つの方法で管理インターフェースにアクセスできます。これらの認証メカニズムの両方の基本は、Red Hat JBoss Enterprise Application Platform セキュリティーアーキテクチャーガイド の デフォルトセキュリティー および Red Hat JBoss Enterprise Application Platform のすぐに使える セクションで説明されています。デフォルトでは、これらのインターフェースへのアクセスは**管理レルム**セキュリティーレルムで設定されます。最初に、ローカルインターフェースが有効になり、JBoss EAP インスタンスを実行するホストマシンへのアクセスになります。リモートアクセスも有効化され、ファイルベースのアイデンティティーストアを使用するように設定されます。デフォルトでは、mgmt-users.properties ファイルを使用してユーザー名とパスワードを保存し、mgmt-groups.properties を使用してユーザーグループ情報を保存します。

ユーザー情報は、**EAP_HOME/bin/** ディレクトリーに格納されている **adduser** スクリプトを使用してこれらのファイルに追加されます。

adduser スクリプトでユーザーを追加するには、以下を実行します。

1. **add-user.sh** または **add-user.bat** コマンドを実行します。
2. 管理ユーザーまたはアプリケーションユーザーを追加するかどうかを選択します。
3. ユーザーを追加するレルムを選択します。デフォルトでは、利用可能なレルムは ManagementRealm と ApplicationRealm のみです。カスタムレルムが追加されると、代わりに手動で名前を入力することができます。
4. プロンプトが表示されたら、希望のユーザー名、パスワード、および任意のロールを入力します。変更は、セキュリティーレルムの各プロパティーファイルに書き込まれます。

3.1.2. LDAP での認証の追加

JBoss EAP は、LDAP 認証を使用した管理インターフェースのセキュア化にも対応しています。LDAP の基本と JBoss EAP での使用は、Red Hat JBoss Enterprise Application Platform 7『セキュリティアーキテクチャー』ガイドの「LDAP」、「LDAP と管理インターフェースの使用」、「LDAP と ManagementRealm の使用」で説明されています。LDAP 認証を使用して管理インターフェースをセキュアにする方法の詳細は、JBoss EAP『How to Configure Identity Management ガイド』の「Securing the Management Interfaces with LDAP」セクションを参照してください。

3.2. 安全なパスワード

3.2.1. パスワード vault

JBoss EAP および関連するアプリケーションの設定には、ユーザー名とパスワードなどの機密情報が必要になります。パスワードをプレーンテキストで設定ファイルに保存する代わりに、パスワード vault 機能を使用してパスワード情報をマスクし、暗号化したキーストアに保存できます。パスワードを保存

したら、JBoss EAP にデプロイされた管理 CLI コマンドまたはアプリケーションに参照を含めることができます。

パスワード vault は Java キーストアをストレージメカニズムとして使用します。パスワード vault はストレージとキーストレージの 2 つで構成されます。Java キーストアは、Vault ストレージで機密文字列を暗号化または復号化するために使用されるキーを保存するために使用されます。

パスワード vault を設定して使用するには、以下の手順が必要です。

1. Java キーストアを設定し、パスワード暗号化用のキーを格納します。
2. パスワード vault の初期化
3. PasswordVault を使用するように Red Hat JBoss Enterprise Application Platform を設定します
4. パスワード Vault に Sensitive 文字列を保存します。
5. 暗号化した機密文字列を設定で使用

3.2.1.1. 1.Java キーストアを設定し、パスワード暗号化用のキーを格納します。

注記

この手順では、Java Runtime Environment (JRE) が提供する keytool ユーティリティーが使用されます。ファイルパスを見つけます。Red Hat Enterprise Linux では、**/usr/bin/keytool** です。

警告

JCEKS キーストアの実装は Java ベンダーごとに異なります。そのため、キーストアは、使用している JDK と同じベンダーの keytool ユーティリティーを使用して生成する必要があります。別のベンダーの JDK で実行している JBoss EAP 7 インスタンスで、あるベンダーの JDK からキーツールによって生成されたキーストアを使用すると、**java.io.IOException: com.sun.crypto.provider.SealedObjectForKeyProtector** の例外が発生します。

キーストアを設定するには、以下の手順を実行します。

1. キーストアおよびその他の暗号化された情報を保存するディレクトリーを作成します。
2. Keytool ユーティリティーで使用するパラメーターを決定します。
3. keytool コマンドを実行します。

1.キーストアおよびその他の暗号化された情報を保存するディレクトリーを作成します。

キーストアと他の重要な情報を格納するディレクトリーを作成します。この手順では、ディレクトリーは **EAP_HOME/vault/** と想定します。このディレクトリーには機密情報が含まれるため、アクセスは、一部のユーザーに制限する必要があります。JBoss EAP 7 を実行しているユーザーアカウントには少なくとも読み書き込みアクセスが必要です。

2.Keytool ユーティリティーで使用するパラメーターを決定します。

以下のパラメーターの値を決めます。

alias

エイリアスは、vault やキーストアに保存されている他のデータの一意識別子です。エイリアスは大文字と小文字を区別しません。

storetype

ストアタイプはキーストアのタイプを指定します。値は、**jceks** が推奨されます。

keyalg

暗号化に使用するアルゴリズム。JRE およびオペレーティングシステムのドキュメントを参照して、利用可能な他の選択肢を確認します。

keysize

暗号化キーのサイズは、ブルートフォースでの暗号解除の難易度に影響します。適切な値の詳細は、キーツールユーティリティーとともに配布されるドキュメントを参照してください。

storepass

storepass の値は、キーストアに対する認証に使用されるパスワードであるため、鍵を読み取ることができます。パスワードは 6 文字以上である必要があります。パスワードは、キーストアへのアクセス時に入力する必要があります。このパラメーターを省略すると、コマンド実行後に keytool ユーティリティーにより入力が求められます。

keypass

Keypass の値は、特定のキーにアクセスするために使用されるパスワードで、storepass パラメーターの値と一致する必要があります。

validity

validity の値は、鍵が有効になる期間 (日数) です。

keystore

keystore の値は、keystore の値が格納される予定のファイルパスおよびファイル名です。キーストアファイルは、データが最初に追加されると作成されます。正しいファイルパス区切り文字が使用されていることを確認します。Red Hat Enterprise Linux および同様のオペレーティングシステムの場合は / (フォワードスラッシュ)、Microsoft Windows Server の場合は \ (バックスラッシュ) を使用します。

Keytool ユーティリティーには、その他の多くのオプションがあります。詳細は、JRE またはオペレーティングシステムのドキュメントを参照してください。

3.keytool コマンドを実行します。

```
$ keytool -genseckey -alias vault -storetype jceks -keyalg AES -keysize 128 -storepass vault22 -
keypass vault22 -validity 730 -keystore EAP_HOME/vault/vault.keystore
```

これにより、**EAP_HOME/vault/vault.keystore** ファイルに作成されたキーストアが作成されます。JBoss EAP では、パスワードなどの暗号化された文字列を保存するために使用されるエイリアス vault とともに単一のキーを保存します。

3.2.1.2. 2.パスワード vault の初期化

パスワード vault は、各パラメーターの値を入力するように求められる対話的な方法、またはすべてのパラメーターの値がコマンドラインで指定される非対話的な方法で初期化できます。各メソッドで同じ結果が表示されるため、どちらかを使用できます。

以下のパラメーターが必要です。

keystore URL (KEYSTORE_URL)

キーストアファイルのファイルシステムパスまたは URI。この例では、**EAP_HOME/vault/vault.keystore** を使用します。

キーストアパスワード (KEYSTORE_PASSWORD)

キーストアのアクセスに使用されるパスワード。

Salt (SALT)

Salt 値は、キーストアの内容を暗号化するために、iteration count (反復カウント) とともに使用される 8 文字のランダムな文字列です。

keystore Alias (KEYSTORE_ALIAS)

キーストア認識されているエイリアス。

Iteration Count (ITERATION_COUNT)

暗号化アルゴリズムの実行回数。

Directory to store encrypted files (ENC_FILE_DIR)

暗号化したファイルを保存するパス。これは通常、パスワード vault を含むディレクトリーです。暗号化されたすべての情報をキーストアと同じ場所に格納することは便利ですが、必須ではありません。このディレクトリーには、制限のあるユーザーのみがアクセスできるようにする必要があります。JBoss EAP 7 を実行しているユーザーアカウントには少なくとも読み書き込みアクセスが必要です。キーストアは、手順 1 で使用したディレクトリーに配置する必要があります。ディレクトリー名の後にはバックスラッシュまたはスラッシュが必要であることに注意してください。正しいファイルパス区切り文字が使用されていることを確認します。Red Hat Enterprise Linux および同様のオペレーティングシステムの場合は / (フォワードスラッシュ)、Microsoft Windows Server の場合は \ (バックスラッシュ) を使用します。

Vault Block (VAULT_BLOCK)

パスワード vault でこのブロックに与えられる名前。

Attribute (ATTRIBUTE)

保存される属性に与えられる名前。

Security Attribute (SEC-ATTR)

パスワード vault に保存されているパスワード。

vault コマンドを非対話的に実行する

パスワード vault コマンドを非対話的に実行するには、**EAP_HOME/bin/** にある **vault** スクリプトを、関連情報のパラメーターを使って呼び出します。

```
vault.sh --keystore KEYSTORE_URL --keystore-password KEYSTORE_PASSWORD --alias
KEYSTORE_ALIAS --vault-block VAULT_BLOCK --attribute ATTRIBUTE --sec-attr SEC-ATTR --
enc-dir ENC_FILE_DIR --iteration ITERATION_COUNT --salt SALT
```

例

```
vault.sh --keystore EAP_HOME/vault/vault.keystore --keystore-password vault22 --alias vault --vault-
block vb --attribute password --sec-attr 0penS3sam3 --enc-dir EAP_HOME/vault/ --iteration 120 --
salt 1234abcd
```

出力

```
==========================================================================
```

```
  JBoss Vault
  JBOSS_HOME: EAP_HOME
  JAVA: java
 ========================================================================
Oct 17, 2014 2:23:43 PM org.picketbox.plugins.vault.PicketBoxSecurityVault init
INFO: PBOX000361: Default Security Vault Implementation Initialized and Ready
Secured attribute value has been stored in vault.
Please make note of the following:
********************************************
Vault Block:vb
Attribute Name:password
Configuration should be done as follows:
VAULT::vb::password::1
********************************************
Vault Configuration in AS7 config file:
********************************************

...
</extensions>
<vault>
  <vault-option name="KEYSTORE_URL" value="EAP_HOME/vault/vault.keystore"/>
  <vault-option name="KEYSTORE_PASSWORD" value="MASK-5dOaAVafCSd"/>
  <vault-option name="KEYSTORE_ALIAS" value="vault"/>
  <vault-option name="SALT" value="1234abcd"/>
  <vault-option name="ITERATION_COUNT" value="120"/>
  <vault-option name="ENC_FILE_DIR" value="EAP_HOME/vault/"/>
</vault><management> ...
********************************************
```

vault コマンドをインタラクティブに実行する

パスワード vault コマンドを対話的に実行するには、以下の手順が必要です。

1. パスワード vault コマンドをインタラクティブに起動します。
2. 要求パラメーターを入力します。
3. マスクされたパスワード情報をメモします。
4. 対話式コンソールを終了します。

1.パスワード vault コマンドをインタラクティブに起動します。

Red Hat Enterprise Linux または同様のオペレーティングシステムでは **EAP_HOME/bin/vault.sh**、Microsoft Windows Server では EAP_HOME\bin\vault.bat を実行します。新しい対話セッションを開始するには、0 (ゼロ) と入力します。

2.要求パラメーターを入力します。

プロンプトに従って必要なパラメーターを入力します。

3.マスクされたパスワード情報をメモします。

マスクされたパスワード、salt、iteration count (反復数) は、標準出力に出力されます。安全な場所にそれらを書き留めておきます。これらのエントリーは、パスワード Vault に追加する必要があります。キーストアファイルおよびこの値にアクセスすると、攻撃者はパスワード Vault の機密情報にアクセスできるようになります。

4.対話式コンソールを終了します。

対話式コンソールを終了するには、2 と入力します。

例: 入出力

```
Please enter a Digit::   0: Start Interactive Session  1: Remove Interactive Session  2: Exit
0
Starting an interactive session
Enter directory to store encrypted files:EAP_HOME/vault/
Enter Keystore URL:EAP_HOME/vault/vault.keystore
Enter Keystore password: vault22
Enter Keystore password again: vault22
Values match
Enter 8 character salt:1234abcd
Enter iteration count as a number (Eg: 44):120
Enter Keystore Alias:vault
Initializing Vault
Oct 17, 2014 12:58:11 PM org.picketbox.plugins.vault.PicketBoxSecurityVault init
INFO: PBOX000361: Default Security Vault Implementation Initialized and Ready
Vault Configuration in AS7 config file:
********************************************
...
</extensions>
<vault>
  <vault-option name="KEYSTORE_URL" value="EAP_HOME/vault/vault.keystore"/>
  <vault-option name="KEYSTORE_PASSWORD" value="MASK-5dOaAVafCSd"/>
  <vault-option name="KEYSTORE_ALIAS" value="vault"/>
  <vault-option name="SALT" value="1234abcd"/>
  <vault-option name="ITERATION_COUNT" value="120"/>
  <vault-option name="ENC_FILE_DIR" value="EAP_HOME/vault/"/>
</vault><management> ...
********************************************
Vault is initialized and ready for use
Handshake with Vault complete
```

+ キーストアパスワードは、設定ファイルおよびデプロイメントで使用するためにマスクされています。さらに、vault は初期化され、使用できる状態になります。

3.2.1.3. 3.PasswordVault を使用するように Red Hat JBoss Enterprise Application Platform を設定します

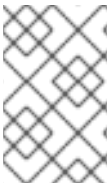
パスワードやその他の機密属性をマスクして設定ファイルで使用できるようにするには、保存および復号化するパスワード vault を JBoss EAP 7 に認識させる必要があります。

以下のコマンドを使用して、JBoss EAP 7 がパスワード vault を使用するように設定できます。

```
/core-service=vault:add( \
vault-options=[ \
("KEYSTORE_URL" => "PATH_TO_KEYSTORE"), \
("KEYSTORE_PASSWORD" => "MASKED_PASSWORD"), \
("KEYSTORE_ALIAS" => "ALIAS"), \
("SALT" => "SALT"), \
("ITERATION_COUNT" => "ITERATION_COUNT"), \
("ENC_FILE_DIR" => "ENC_FILE_DIR")])
```

CLI コマンドの例

```
/core-service=vault:add( \
vault-options=[ \
("KEYSTORE_URL" => "EAP_HOME/vault/vault.keystore"), \
("KEYSTORE_PASSWORD" => "MASK-5dOaAVafCSd"), \
("KEYSTORE_ALIAS" => "vault"), \
("SALT" => "1234abcd"), \
("ITERATION_COUNT" => "120"), \
("ENC_FILE_DIR" => "EAP_HOME/vault/")])
```



注記

Microsoft Windows Server を使用している場合は、バックスラッシュ (\\) を 1 つではなく 2 つ使用します。例: **C:\\data\\vault\\vault.keystore**これは、単一のバックスラッシュ (\) が文字エスケープに使用されるためです。

これで、JBoss EAP 6 は、PasswordVault に保存されているマスクされた文字列を復号化するように設定されました。

3.2.1.4. 4.パスワード Vault に Sensitive 文字列を保存します。

プレーンテキストの設定ファイルにパスワードやその他の機密文字列を含めると、セキュリティーリスクが伴います。セキュリティ上の理由からも、これらの文字列はパスワード vault に保存します。パスワード vault では、これらの文字列は、マスク化した形式で設定ファイル、管理 CLI コマンド、およびアプリケーションで参照できます。

機密文字列は、ツールが各パラメーターの値を要求する場合には、パスワード Vault に対話形式で格納することができます。あるいは、コマンドラインですべてのパラメーターの値が提供される非対話形式で保存することもできます。各メソッドで同じ結果が表示されるため、どちらかを使用できます。これらのメソッドは、両者とも **vault** スクリプトで呼び出しされます。

機密性の高い文字列を非対話的に保存する

パスワード vault コマンドを非対話的に実行するには、**EAP_HOME/bin/** にある **vault** スクリプトを、関連情報のパラメーターを使って呼び出します。

```
EAP_HOME/bin/vault.sh --keystore KEYSTORE_URL --keystore-password
KEYSTORE_PASSWORD --alias KEYSTORE_ALIAS --vault-block VAULT_BLOCK --attribute
ATTRIBUTE --sec-attr SEC-ATTR --enc-dir ENC_FILE_DIR --iteration ITERATION_COUNT --salt
SALT
```

注記

キーストアパスワードは、マスク形式ではなく、プレーンテキスト形式で指定する必要があります。

例

```
EAP_HOME/bin/vault.sh --keystore EAP_HOME/vault/vault.keystore --keystore-password vault22 --
alias vault --vault-block vb --attribute password --sec-attr 0penS3sam3 --enc-dir EAP_HOME/vault/ --
iteration 120 --salt 1234abcd
```

出力

-

```
 ========================================================================
  JBoss Vault
  JBOSS_HOME: EAP_HOME
  JAVA: java
 ========================================================================
Oct 22, 2014 9:24:43 AM org.picketbox.plugins.vault.PicketBoxSecurityVault init
INFO: PBOX000361: Default Security Vault Implementation Initialized and Ready
Secured attribute value has been stored in vault.
Please make note of the following:
********************************************
Vault Block:vb
Attribute Name:password
Configuration should be done as follows:
VAULT::vb::password::1
********************************************
Vault Configuration in AS7 config file:
********************************************
...
</extensions>
<vault>
  <vault-option name="KEYSTORE_URL" value="EAP_HOME/vault/vault.keystore"/>
  <vault-option name="KEYSTORE_PASSWORD" value="vault22"/>
  <vault-option name="KEYSTORE_ALIAS" value="vault"/>
  <vault-option name="SALT" value="1234abcd"/>
  <vault-option name="ITERATION_COUNT" value="120"/>
  <vault-option name="ENC_FILE_DIR" value="EAP_HOME/vault/vault/"/>
</vault><management> ...
********************************************
```

vault スクリプトを呼び出した後、メッセージは標準出力に出力されます。このとき、vault ブロック、属性名、マスクされた文字列、設定で文字列を使用することについてのアドバイスが表示されます。この情報は、安全な場所にメモしておいてください。出力例を以下に示します。

```
Vault Block:vb
Attribute Name:password
Configuration should be done as follows:
VAULT::vb::password::1
```

対話式での機密文字列の保存

パスワード vault コマンドを対話的に実行するには、以下の手順が必要です。

1. パスワード vault コマンドをインタラクティブに起動します。
2. 要求パラメーターを入力します。
3. 機密文字列に関するパラメーター入力を行います。
4. マスクされた文字列についての情報を書き留めておきます。
5. 対話式コンソールを終了します。

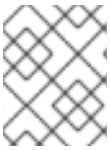
1.パスワード vault コマンドをインタラクティブに起動します。

オペレーティングシステムのコマンドラインインターフェースを起動し、**EAP_HOME/bin/vault.sh** (Red Hat Enterprise Linux および同様のオペレーティングシステム) または EAP_HOME\bin\vault.bat

(Microsoft Windows Server 上) を実行します。新しい対話セッションを開始するには、0 (ゼロ) と入力します。

2.要求パラメーターを入力します。

プロンプトに従って必要なパラメーターを入力します。これらの値は、パスワード vault の作成時に提供された値と一致している必要があります。



注記

キーストアパスワードは、マスク形式ではなく、プレーンテキスト形式で指定する必要があります。

3.機密文字列に関するパラメーター入力を行います。

機密文字列の保存を開始する場合は 0 (ゼロ) を入力します。プロンプトに従って必要なパラメーターを入力します。

4.マスクされた文字列についての情報を書き留めておきます。

メッセージは標準出力に出力され、vault ブロック、属性名、マスクされた文字列、設定の文字列の使用に関するアドバイスが表示します。この情報は、安全な場所にメモしておいてください。出力例を以下に示します。

```
Vault Block:ds_Example1
Attribute Name:password
Configuration should be done as follows:
VAULT::ds_Example1::password::1
```

5.対話式コンソールを終了します。

対話式コンソールを終了するには、2 と入力します。

例: 入出力

```
 ========================================================================
 JBoss Vault
 JBOSS_HOME: EAP_HOME/jboss-eap-6.4
 JAVA: java
 ========================================================================
 *********************************
 ****  JBoss Vault  ***************
 *********************************
Please enter a Digit::   0: Start Interactive Session  1: Remove Interactive Session  2: Exit
0
Starting an interactive session
Enter directory to store encrypted files:EAP_HOME/vault/
Enter Keystore URL:EAP_HOME/vault/vault.keystore
Enter Keystore password:
Enter Keystore password again:
Values match
Enter 8 character salt:1234abcd
Enter iteration count as a number (Eg: 44):120
Enter Keystore Alias:vault
Initializing Vault
Oct 21, 2014 11:20:49 AM org.picketbox.plugins.vault.PicketBoxSecurityVault init
```

```
INFO: PBOX000361: Default Security Vault Implementation Initialized and Ready
Vault Configuration in AS7 config file:
 ************************************************
...
</extensions>
<vault>
  <vault-option name="KEYSTORE_URL" value="EAP_HOME/vault/vault.keystore"/>
  <vault-option name="KEYSTORE_PASSWORD" value="MASK-5dOaAVafCSd"/>
  <vault-option name="KEYSTORE_ALIAS" value="vault"/>
  <vault-option name="SALT" value="1234abcd"/>
  <vault-option name="ITERATION_COUNT" value="120"/>
  <vault-option name="ENC_FILE_DIR" value="EAP_HOME/vault/"/>
</vault><management> ...
 ************************************************
Vault is initialized and ready for use
Handshake with Vault complete
Please enter a Digit::  0: Store a secured attribute  1: Check whether a secured attribute exists  2:
Remove secured attribute  3: Exit
0
Task: Store a secured attribute
Please enter secured attribute value (such as password):
Please enter secured attribute value (such as password) again:
Values match
Enter Vault Block:ds_Example1
Enter Attribute Name:password
Secured attribute value has been stored in vault.
Please make note of the following:
 ************************************************
Vault Block:ds_Example1
Attribute Name:password
Configuration should be done as follows:
VAULT::ds_Example1::password::1
 ************************************************
Please enter a Digit::  0: Store a secured attribute  1: Check whether a secured attribute exists  2:
Remove secured attribute  3: Exit
```

3.2.1.5. 5.暗号化した機密文字列を設定で使用

暗号化されている機密文字列は、マスクされた形式で設定ファイルまたは管理 CLI コマンド内で使用でき、式の指定も可能です。

特定のサブシステム内で式が許可されているかどうかを確認するには、そのサブシステムに対して以下の管理 CLI コマンドを実行します。

```
/subsystem=SUBSYSTEM:read-resource-description(recursive=true)
```

このコマンドの実行の出力から、**expressions-allowed** パラメーターの値を探します。true の場合は、このサブシステムの設定内で式を使用できます。

以下の構文を使用して、プレーンテキスト文字列をマスクされたフォームに置き換えます。

```
${VAULT::VAULT_BLOCK::ATTRIBUTE_NAME::MASKED_STRING}
```

例 - マスクされた形式のパスワードを使用したデータソースの定義

-

```
...
  <subsystem xmlns="urn:jboss:domain:datasources:1.0">
    <datasources>
      <datasource jndi-name="java:jboss/datasources/ExampleDS" enabled="true" use-java-
context="true" pool-name="H2DS">
        <connection-url>jdbc:h2:mem:test;DB_CLOSE_DELAY=-1</connection-url>
        <driver>h2</driver>
        <pool></pool>
        <security>
          <user-name>sa</user-name>
          <password>${VAULT::ds_ExampleDS::password::1}</password>
        </security>
      </datasource>
      <drivers>
        <driver name="h2" module="com.h2database.h2">
          <xa-datasource-class>org.h2.jdbcx.JdbcDataSource</xa-datasource-class>
        </driver>
      </drivers>
    </datasources>
  </subsystem>
...
```

3.2.1.6. アプリケーションで暗号化した機密文字列の使用

パスワード vault に保存されている暗号化された文字列は、アプリケーションのソースコードで使用できます。以下の例は、サーブレットのソースコードの抜粋です。これは、プレーンテキストのパスワードではなく、データソース定義でのマスクされたパスワードの使用を示しています。プレーンテキストバージョンはコメントアウトされているため、違いがわかります。

vault されたパスワードを使用するサーブレット

```
@DataSourceDefinition(
        name = "java:jboss/datasources/LoginDS",
        user = "sa",
        password = "VAULT::DS::thePass::1",
        className = "org.h2.jdbcx.JdbcDataSource",
        url = "jdbc:h2:tcp://localhost/mem:test"
)
/*old (plaintext) definition
@DataSourceDefinition(
        name = "java:jboss/datasources/LoginDS",
        user = "sa",
        password = "sa",
        className = "org.h2.jdbcx.JdbcDataSource",
        url = "jdbc:h2:tcp://localhost/mem:test"
)*/
```

3.2.1.7. 機密文字列がパスワード vault 内にあるかどうかを確認します。

パスワード vault に機密文字列を保存または使用する前に、その文字列がすでに保存されているかどうかを確認すると便利です。

このチェックは、ユーザーに各パラメーターの値の入力を求める対話形式または、コマンドラインですべてのパラメーターの値が提供される非対話形式のいずれかで実行できます。各メソッドで同じ結果が表示されるため、どちらかを使用できます。これらのメソッドは、両者とも **vault** スクリプトで呼び出

しされます。

機密性の高い文字列を非対話的にチェックする

このメソッドを使用して、すべてのパラメーターの値を一度に提供します。すべてのパラメーターの説明は、「パスワード vault の初期化」を参照してください。

パスワード vault コマンドを非対話的に実行するには、**EAP_HOME/bin/** にある **vault** スクリプトを、関連情報のパラメーターを使って呼び出します。

```
EAP_HOME/bin/vault.sh --keystore KEYSTORE_URL --keystore-password
KEYSTORE_PASSWORD --alias KEYSTORE_ALIAS --check-sec-attr --vault-block VAULT_BLOCK
--attribute ATTRIBUTE --enc-dir ENC_FILE_DIR --iteration ITERATION_COUNT --salt SALT
```

プレースホルダーの値を実際の値に置き換えます。パラメーター KEYSTORE_URL、KEYSTORE_PASSWORD、および KEYSTORE_ALIAS の値は、パスワード vault の作成時に提供された値と一致している必要があります。

注記

キーストアパスワードは、マスク形式ではなく、プレーンテキスト形式で指定する必要があります。

機密文字列が指定された vault ブロックに保存されると、以下のメッセージが表示されます。

```
Password already exists.
```

値が指定のブロックに保存されていない場合は、以下のメッセージが表示されます。

```
Password doesn't exist.
```

対話式での機密文字列の確認

パスワード vault コマンドを対話的に実行するには、以下の手順が必要です。

1. パスワード vault コマンドをインタラクティブに起動します。
2. 要求パラメーターを入力します。

1.パスワード vault コマンドをインタラクティブに起動します。

Red Hat Enterprise Linux または同様のオペレーティングシステムでは **EAP_HOME/bin/vault.sh**、Microsoft Windows Server では EAP_HOME\bin\vault.bat を実行します。新しい対話セッションを開始するには、0 (ゼロ) と入力します。

2.要求パラメーターを入力します。

プロンプトに従って必要な認証パラメーターを入力します。これらの値は、パスワード vault の作成時に提供された値と一致している必要があります。

注記

認証が求められたときは、キーストアパスワードはマスク形式ではなく、プレーンテキスト形式で指定する必要があります。

- 「1」を入力して セキュリティー保護された属性が存在するかどうかを確認します。
- 機密文字列を保存する vault ブロックの名前を入力します。
- チェックする機密文字列の名前を入力します。

機密文字列が指定された vault ブロックに保存されている場合は、以下のような確認メッセージが表示されます。

```
A value exists for (VAULT_BLOCK, ATTRIBUTE)
```

機密文字列が指定ブロックに保存されていない場合は、以下のようなメッセージが出力されます。

```
No value has been store for (VAULT_BLOCK, ATTRIBUTE)
```

例: 対話式での機密文字列の確認

```
 ===========================================================================
  JBoss Vault
  JBOSS_HOME: EAP_HOME
  JAVA: java
 ===========================================================================
 *********************************
 ****  JBoss Vault  ***************
 *********************************
Please enter a Digit::   0: Start Interactive Session  1: Remove Interactive Session  2: Exit
0
Starting an interactive session
Enter directory to store encrypted files:EAP_HOME/vault
Enter Keystore URL:EAP_HOME/vault/vault.keystore
Enter Keystore password:
Enter Keystore password again:
Values match
Enter 8 character salt:1234abcd
Enter iteration count as a number (Eg: 44):120
Enter Keystore Alias:vault
Initializing Vault
Oct 22, 2014 12:53:56 PM org.picketbox.plugins.vault.PicketBoxSecurityVault init
INFO: PBOX000361: Default Security Vault Implementation Initialized and Ready
Vault Configuration in AS7 config file:
 ********************************************
...
</extensions>
<vault>
  <vault-option name="KEYSTORE_URL" value="EAP_HOME/vault/vault.keystore"/>
  <vault-option name="KEYSTORE_PASSWORD" value="MASK-5dOaAVafCSd"/>
  <vault-option name="KEYSTORE_ALIAS" value="vault"/>
  <vault-option name="SALT" value="1234abcd"/>
  <vault-option name="ITERATION_COUNT" value="120"/>
  <vault-option name="ENC_FILE_DIR" value="EAP_HOME/vault/"/>
</vault><management> ...
 ********************************************
Vault is initialized and ready for use
Handshake with Vault complete
Please enter a Digit::  0: Store a secured attribute  1: Check whether a secured attribute exists  2:
```

```
Remove secured attribute  3: Exit
1
Task: Verify whether a secured attribute exists
Enter Vault Block:vb
Enter Attribute Name:password
A value exists for (vb, password)
Please enter a Digit::  0: Store a secured attribute  1: Check whether a secured attribute exists  2:
Remove secured attribute  3: Exit
```

3.2.1.8. パスワード vault からの機密文字列の削除

セキュリティー上の理由から、機密文字列が不要になった場合は、パスワード vault から削除することが推奨されます。たとえば、アプリケーションの使用を停止する場合、データソース定義で使用される機密文字列を同時に削除する必要があります。

重要

前提条件として、パスワード vault から機密文字列を削除するには、JBoss EAP の設定で使用されるかどうかを確認します。

この操作は、ユーザーに各パラメーターの値の入力を求める対話形式または、コマンドラインですべてのパラメーターの値が提供される非対話形式のいずれかで実行できます。各メソッドで同じ結果が表示されるため、どちらかを使用できます。これらのメソッドは、両者とも **vault** スクリプトで呼び出しされます。

機密性の高い文字列を非対話的に削除する

このメソッドを使用して、すべてのパラメーターの値を一度に提供します。すべてのパラメーターの説明は、「パスワード vault の初期化」を参照してください。

パスワード vault コマンドを非対話的に実行するには、**EAP_HOME/bin/** にある **vault** スクリプトを、関連情報のパラメーターを使って呼び出します。

```
EAP_HOME/bin/vault.sh --keystore KEYSTORE_URL --keystore-password
KEYSTORE_PASSWORD --alias KEYSTORE_ALIAS --remove-sec-attr --vault-block
VAULT_BLOCK --attribute ATTRIBUTE --enc-dir ENC_FILE_DIR --iteration ITERATION_COUNT --
salt SALT
```

プレースホルダーの値を実際の値に置き換えます。パラメーター KEYSTORE_URL、KEYSTORE_PASSWORD、および KEYSTORE_ALIAS の値は、パスワード vault の作成時に提供された値と一致している必要があります。

注記

キーストアパスワードは、マスク形式ではなく、プレーンテキスト形式で指定する必要があります。

機密文字列が正常に削除されると、以下のような確認メッセージが表示されます。

```
Secured attribute [VAULT_BLOCK::ATTRIBUTE] has been successfully removed from vault
```

機密文字列が削除されない場合は、以下のようなメッセージが表示されます。

```
Secured attribute [VAULT_BLOCK::ATTRIBUTE] was not removed from vault, check whether it exist
```

出力例

```
 ./vault.sh --keystore EAP_HOME/vault/vault.keystore --keystore-password vault22 --alias vault --
remove-sec-attr --vault-block vb --attribute password --enc-dir EAP_HOME/vault/ --iteration 120 --salt
1234abcd
 ================================================================================
  JBoss Vault
  JBOSS_HOME: EAP_HOME
  JAVA: java
 ================================================================================
Dec 23, 2014 1:54:24 PM org.picketbox.plugins.vault.PicketBoxSecurityVault init
INFO: PBOX000361: Default Security Vault Implementation Initialized and Ready
Secured attribute [vb::password] has been successfully removed from vault
```

対話式での機密文字列の削除

パスワード vault コマンドを対話的に実行するには、以下の手順が必要です。

1. パスワード vault コマンドをインタラクティブに起動します。
2. 要求パラメーターを入力します。

1.パスワード vault コマンドをインタラクティブに起動します。

Red Hat Enterprise Linux または同様のオペレーティングシステムでは **EAP_HOME/bin/vault.sh**、Microsoft Windows Server では EAP_HOME\bin\vault.bat を実行します。新しい対話セッションを開始するには、0 (ゼロ) と入力します。

2.要求パラメーターを入力します。

プロンプトに従って必要な認証パラメーターを入力します。これらの値は、パスワード vault の作成時に提供された値と一致している必要があります。

注記

認証が求められたときは、キーストアパスワードはマスク形式ではなく、プレーンテキスト形式で指定する必要があります。

- 2 と入力して、セキュアな属性を削除するオプションを選択します。
- 機密文字列を保存する vault ブロックの名前を入力します。
- 削除する機密文字列の名前を入力します。

機密文字列が正常に削除されると、以下のような確認メッセージが表示されます。

```
Secured attribute [VAULT_BLOCK::ATTRIBUTE] has been successfully removed from vault
```

機密文字列が削除されない場合は、以下のようなメッセージが表示されます。

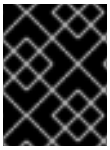
```
Secured attribute [VAULT_BLOCK::ATTRIBUTE] was not removed from vault, check whether it exist
```

出力例

```
 ********************************
 ****  JBoss Vault  ***************
 ********************************
Please enter a Digit::   0: Start Interactive Session  1: Remove Interactive Session  2: Exit
0
Starting an interactive session
Enter directory to store encrypted files:EAP_HOME/vault/
Enter Keystore URL:EAP_HOME/vault/vault.keystore
Enter Keystore password:
Enter Keystore password again:
Values match
Enter 8 character salt:1234abcd
Enter iteration count as a number (Eg: 44):120
Enter Keystore Alias:vault
Initializing Vault
Dec 23, 2014 1:40:56 PM org.picketbox.plugins.vault.PicketBoxSecurityVault init
INFO: PBOX000361: Default Security Vault Implementation Initialized and Ready
Vault Configuration in configuration file:
 ********************************************
...
</extensions>
<vault>
  <vault-option name="KEYSTORE_URL" value="EAP_HOME/vault/vault.keystore"/>
  <vault-option name="KEYSTORE_PASSWORD" value="MASK-5dOaAVafCSd"/>
  <vault-option name="KEYSTORE_ALIAS" value="vault"/>
  <vault-option name="SALT" value="1234abcd"/>
  <vault-option name="ITERATION_COUNT" value="120"/>
  <vault-option name="ENC_FILE_DIR" value="EAP_HOME/vault/"/>
</vault><management> ...
 ********************************************
Vault is initialized and ready for use
Handshake with Vault complete
Please enter a Digit::  0: Store a secured attribute  1: Check whether a secured attribute exists  2:
Remove secured attribute  3: Exit
2
Task: Remove secured attribute
Enter Vault Block:vb
Enter Attribute Name:password
Secured attribute [vb::password] has been successfully removed from vault
```

3.2.1.9. Red Hat JBoss Enterprise Application Platform がパスワード vault のカスタム実装を使用するように設定する

指定したパスワード vault 実装を使用することに加え、SecurityVault のカスタム実装を使用することもできます。



重要

前提条件として、パスワード vault が初期化されていることを確認します。詳細については 、パスワードボールトの初期化 を参照してください。

パスワード vault にカスタム実装を使用するには、以下を実行します。

1. インターフェース **SecurityVault** を実装するクラスを作成します。

2. 直前の手順からクラスを含むモジュールを作成し、インターフェースが **SecurityVault** である **org.picketbox** で依存関係を指定します。

3. 以下の属性で vault 要素を追加して、JBoss EAP 設定でカスタムパスワード vault を有効にします。

   - **code**: SecurityVault を実装するクラスの完全修飾名。
   - **module**: カスタムクラスが含まれるモジュールの名前。

オプションで、**vault-options** パラメーターを使用して、パスワード vault のカスタムクラスを初期化できます。

例: vault-options パラメーターを使用したカスタムクラスの使用

```
/core-service=vault:add( \
code="custom.vault.implementation.CustomSecurityVault", \
module="custom.vault.module", vault-options=[ \
("KEYSTORE_URL" => "PATH_TO_KEYSTORE"), \
("KEYSTORE_PASSWORD" => "MASKED_PASSWORD"), ("KEYSTORE_ALIAS" => "ALIAS"), \
("SALT" => "SALT"),("ITERATION_COUNT" => "ITERATION_COUNT"), \
("ENC_FILE_DIR" => "ENC_FILE_DIR")])
```

3.2.1.10. 外部ソースからのキーストアパスワードの取得

EXT、EXTC、CMD CMDC、または CLASS メソッドは、Java キーストアパスワードの取得について vault 設定で使用できます。

```
<vault-option name="KEYSTORE_PASSWORD" value="[here]"
```

この方法を以下で説明します。

{EXT}...

これは、そのまま使えるコマンドを参照します。ここでの は、実際のコマンドです。例: **{EXT}/usr/bin/getmypassword --section 1 --query company**。**/usr/bin/getmypassword** コマンドを実行します。このコマンドは、標準出力にパスワードを表示し、セキュリティー vault のキーストアのパスワードとして使用します。この例では --section 1 と --query company のオプションを使用しています。

{EXTC[:expiration_in_millis]}...

実際のコマンドを参照します。ここでの、... は、プラットフォームコマンドを実行するために **Runtime.exec(String)** メソッドに渡される、実際のコマンドラインです。コマンド出力の最初の行がパスワードとして使用されます。EXTC バリアントは expiration_in_millis ミリ秒のパスワードをキャッシュします。デフォルトのキャッシュの有効期限は 0 = infinity です。例: {EXTC:120000}/usr/bin/getmypassword --section 1 --query company は、キャッシュに /usr/bin/getmypassword が含まれるかどうかを検証します。出力が含まれる場合は、これを使用します。出力がない場合は、コマンドを実行してこれをキャッシュに出力して使用します。この例では、キャッシュは 2 分 (120000 ミリ秒) で期限切れになります。

{CMD}... or {CMDC[:expiration_in_millis]}...

一般的なコマンドは , (コンマ) で区切られた文字列です。最初の部分は実際のコマンドで、追加の部分はパラメーターを表します。コンマにバックスラッシュを付けることで、パラメーターの一部として維持することができます。例: {CMD}/usr/bin/getmypassword,--section,1,--query,company

{CLASS[@jboss_module_spec]}classname[:ctorargs]

[:ctorargs] は、クラス名から : (コロン) によって区切られるオプションの文字列です。これは、ク

ラス名 **ctor** に渡されます。ctorargs は文字列のコンマ区切りの一覧です。例: {CLASS@org.test.passwd}org.test.passwd.ExternamPassworProviderこの例では、org.test.passwd.ExternamPassworProvider クラスが org.test.passwd モジュールからロードされ、toCharArray() メソッドを使用してパスワードを取得します。toCharArray() が利用できない場合は、toString() メソッドが使用されます。org.test.passwd.ExternamPassworProvider クラスにはデフォルトのコンストラクターが必要です。

3.3. ロールベースのアクセス制御

ロールベースのアクセス制御の基本については、Red Hat JBoss Enterprise Application Platform セキュリティーアーキテクチャーガイド の ロールベースのアクセス制御 と 管理インターフェイスへの RBAC の追加 セクションで説明しています。

3.3.1. ロールベースのアクセス制御の有効化

デフォルトでは、Role-Based Access Control (RBAC) システムが無効になっています。有効にするには、provider 属性を **simple** から **rbac** に変更します。provider は、management 要素の access-control 要素の属性です。これは、管理 CLI を使用するか、サーバーがオフラインの場合にはサーバー設定 XML ファイルを編集して実行できます。稼働中のサーバーで RBAC を無効化または有効化した場合、サーバー設定をリロードして変更を反映する必要があります。

一度有効にすると、無効化できるのは Administrator または SuperUser ロールのユーザーのみとなります。デフォルトでは、サーバーと同じマシンで実行される場合、管理 CLI は SuperUser ロールとして実行されます。

管理 CLI で RBAC を有効にするには、アクセス承認リソースの **write-attribute** 操作で provider 属性を **rbac** に設定します。

RBAC を有効化する CLI

```
/core-service=management/access=authorization:write-attribute( \
name=provider, value=rbac)
```

管理 CLI で RBAC を無効化にするには、アクセス承認リソースの **write-attribute** 操作で provider 属性を **simple** に設定します。

RBAC を無効にする CLI

```
/core-service=management/access=authorization:write-attribute( \
name=provider, value=simple)
```

サーバーがオフラインの場合、XML 設定を編集して RBAC を有効または無効にすることができます。これを行うには、管理要素の access-control 要素の provider 属性を編集します。この値を **rbac** に設定して有効にし、**simple** で無効にします。

XML の例

```
<management>
  <access-control provider="rbac">
    <role-mapping>
      <role name="SuperUser">
        <include>
          <user name="$local"/>
        </include>
```

```
      </role>
    </role-mapping>
  </access-control>
</management>
```

3.3.2. Permission Combination Policy の変更

Permission Combination Policy は、ユーザーが複数のロールが割り当てられているかどうかの判断方法を決定します。permissive または reject に設定できます。デフォルトは **permissive** です。

permissive に設定すると、アクションを許可するユーザーにロールが割り当てられていると、そのアクションが許可されます。

rejecting に設定すると、複数のロールがユーザーに割り当てられている場合、アクションは許可されません。つまり、このポリシーが rejecting に設定されていると、各ユーザーには単一のロールのみを割り当てる必要があります。ポリシーが rejecting に設定されている場合、複数のロールを持つユーザーは管理コンソールまたは管理 CLI を使用できません。

Permission Combination Policy は、**permission-combination-policy** 属性を **permissive** または **rejecting** のいずれかに設定して構成します。これは、管理 CLI を使用するか、サーバーがオフラインの場合にはサーバー設定 XML ファイルを編集して実行できます。Permission-combination-policy 属性は access-control 要素の一部で、access-control 要素は **management** 要素にあります。

Permission Combination Policy の設定

アクセス承認リソースの write-attribute 操作を使用して、permission-combination-policy 属性を必要なポリシー名に設定します。

```
/core-service=management/access=authorization:write-attribute( \
name=permission-combination-policy, value=POLICYNAME)
```

有効なポリシー名は **rejecting** および **permissive** になります。

CLI の例

```
/core-service=management/access=authorization:write-attribute( \
name=permission-combination-policy, value=rejecting)
```

サーバーがオフラインの場合、XML 設定を編集してパーミッションの組み合わせポリシー (permission combination policy) 値を変更できます。これを行うには、access-control 要素の permission-combination-policy 属性を編集します。

XML の例

```
<access-control provider="rbac" permission-combination-policy="rejecting">
  <role-mapping>
    <role name="SuperUser">
      <include>
        <user name="$local"/>
      </include>
    </role>
  </role-mapping>
</access-control>
```

3.3.3. ロールの管理

RBAC (Role-Based Access Control) を有効にすると、管理ユーザーが許可されている内容は、ユーザーが割り当てられているロールによって決まります。JBoss EAP 6 は、ユーザーおよびグループメンバーシップの両方に基づた包含と除外のシステムを使用して、ユーザーが所属するロールを決定します。

ユーザーは、以下の場合にロールに割り当てられると見なされます。

- ロールに含めるユーザーとして一覧表示される、または
- ロールに含まれるように一覧表示されるグループのメンバー。

また、ユーザーが以下を実行しない場合、ユーザーはロールに割り当てられると見なされます。

- ロールから除外するユーザーとして一覧、または
- ロールから除外される一覧のグループのメンバーです。

除外は包含よりも優先されます。

ユーザーおよびグループのロール包含および除外の設定は、管理コンソールと管理 CLI の両方を使用して設定できます。

この設定を実行できるのは、**SuperUser** または **Administrator** ロールのユーザーのみです。

3.3.3.1. 管理 CLI を用いたユーザーロール割り当ての設定

ユーザーおよびグループのロールへのマッピングの設定は、**role-mapping** 要素として **/core-service=management/access=authorization** に位置します。

この設定を実行できるのは、**SuperUser** または **Administrator** ロールのユーザーのみです。

ロール割り当て設定の表示

read-children-names 操作を使用して、設定されたロールの完全なリストを取得します。

```
/core-service=management/access=authorization:read-children-names(child-type=role-mapping)
{
    "outcome" => "success",
    "result" => [
        "Administrator",
        "Deployer",
        "Maintainer",
        "Monitor",
        "Operator",
        "SuperUser"
    ]
}
```

指定した role-mapping の **read-resource** 操作を使用して、特定のロールの完全な詳細を取得します。

```
/core-service=management/access=authorization/role-mapping=ROLENAME:read-
resource(recursive=true)
{
    "outcome" => "success",
    "result" => {
```

```
        "include-all" => false,
        "exclude" => undefined,
        "include" => {
            "user-theboss" => {
                "name" => "theboss",
                "realm" => undefined,
                "type" => "USER"
            },
            "user-harold" => {
                "name" => "harold",
                "realm" => undefined,
                "type" => "USER"
            },
            "group-SysOps" => {
                "name" => "SysOps",
                "realm" => undefined,
                "type" => "GROUP"
            }
        }
    }
}
```

新規ロールの追加

この手順では、ロールの role-mapping エントリーを追加する方法を説明します。これは、ロールを設定する前に行う必要があります。

add 操作で、新しいロール設定を追加します。

```
/core-service=management/access=authorization/role-mapping=ROLENAME:add
```

- ROLENAME は、新しいマッピングの対象となるロールの名前です (例: 監査人)。

CLI の例

```
/core-service=management/access=authorization/role-mapping=Auditor:add
```

ロールに include されるユーザーの追加

この手順では、ユーザーをロールの include されたリストに追加する方法を説明します。

ロールの設定が行われていない場合は、そのロールの role-mapping エントリーを最初に実行する必要があります。

add 操作を使用して、ロール、追加一覧にユーザーエントリーを追加します。

```
/core-service=management/access=authorization/role-mapping=ROLENAME/include= \
ALIAS:add(name=USERNAME, type=USER)
```

- ROLENAME は、設定されているロールの名前です。(例: 監査人)
- ALIAS は、このマッピングの一意の名前です。Red Hat は、user-USERNAME (例: user-max) などのエイリアスの命名規則を使用することを推奨します。
- username は、包含リストに追加されるユーザーの名前です。

CLI の例

```
/core-service=management/access=authorization/role-mapping=Auditor/include= \
user-max:add(name=max, type=USER)
```

ロールに exclude されるユーザーの追加

この手順では、ロールの除外リストにユーザーを追加する方法を説明します。

ロールの設定が行われていない場合は、そのロールの role-mapping エントリーを最初に実行する必要があります。

add 操作を使用して、ロールの除外リストにユーザーエントリーを追加します。

```
/core-service=management/access=authorization/role-mapping=ROLENAME/exclude= \
ALIAS:add(name=USERNAME, type=USER)
```

- ROLENAME は、設定されているロールの名前です。(例: 監査人)
- USERNAME は、除外リストに追加されているユーザーの名前です。
- ALIAS は、このマッピングの一意の名前です。Red Hat は、user-USERNAME (例: user-max) などのエイリアスの命名規則を使用することを推奨します。

CLI の例

```
/core-service=management/access=authorization/role-mapping=Auditor/exclude= \
user-max:add(name=max, type=USER)
```

ユーザーロールの include 設定の削除

この手順では、ロールマッピングからユーザー包含エントリーを削除する方法を説明します。

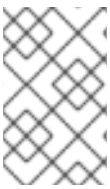
remove 操作を使用してエントリーを削除します。

```
/core-service=management/access=authorization/role-mapping=ROLENAME/include= \
ALIAS:remove
```

- ROLENAME は、設定されているロールの名前です (例: 監査人)
- ALIAS は、このマッピングの一意の名前です。Red Hat は、user-USERNAME (例: user-max) などのエイリアスの命名規則を使用することを推奨します。

CLI の例

```
/core-service=management/access=authorization/role-mapping=Auditor/include= \
user-max:remove
```



注記

包含の一覧からユーザーを削除しても、ユーザーはシステムから削除されません。また、ロールがそのユーザーに割り当てられないようにします。ロールはグループメンバーシップに基づいて依然として割り当てられる可能性があります。

ユーザーロールの exclude 設定の削除

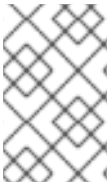
この手順では、ロールマッピングからユーザー除外エントリーを削除する方法を説明します。

削除操作でエントリーを削除します。

```
/core-service=management/access=authorization/role-mapping=ROLENAME/exclude= \
ALIAS:remove
```

- ROLENAME は、設定されているロールの名前です。(例: 監査人)
- ALIAS は、このマッピングの一意の名前です。Red Hat は、user-USERNAME (例: user-max) などのエイリアスの命名規則を使用することを推奨します。

```
/core-service=management/access=authorization/role-mapping=Auditor/exclude= \
user-max:remove
```



注記

除外の一覧からユーザーを削除しても、ユーザーはシステムから削除されません。また、そのロールが必ずそのユーザーに割り当てられるようになるわけではありません。依然としてロールはグループメンバーシップに基づいて除外される可能性があります。

3.3.4. ロールおよびユーザーグループ

mgmt-users.properties ファイルまたは LDAP サーバーのいずれかを使用して認証されたユーザーは、ユーザーグループのメンバーになることができます。ユーザーグループは、1 人以上のユーザーに割り当てることのできる任意のラベルです。

RBAC システムは、所属するユーザーグループに応じて、自動的にロールをユーザーに割り当てるように設定できます。また、グループメンバーシップに基づいてロールからユーザーを除外することもできます。

mgmt-users.properties ファイルを使用する場合、グループ情報は **mgmt-groups.properties** ファイルに保存されます。LDAP を使用する場合、グループ情報は LDAP サーバーに保存され、LDAP サーバーの責任者によって維持されます。

3.3.5. 管理 CLI を用いたグループロール割り当ての設定

ロールに含まれる、またはロールから除外されるグループは、管理コンソールおよび管理 CLI で設定できます。このトピックでは、管理 CLI の使用についてのみ説明します。

ユーザーおよびグループのロールへのマッピングの設定は、**role-mapping** 要素として **/core-service=management/access=authorization** の管理 API に位置します。

この設定を実行できるのは、**SuperUser** または **Administrator** ロールのユーザーのみです。

グループロール割り当て設定の表示

read-children-names 操作を使用して、設定されたロールの完全なリストを取得します。

```
/core-service=management/access=authorization:read-children-names(child-type=role-mapping)
{
    "outcome" => "success",
    "result" => [
```

```
        "Administrator",
        "Deployer",
        "Maintainer",
        "Monitor",
        "Operator",
        "SuperUser"
    ]
}
```

指定した role-mapping の **read-resource** 操作を使用して、特定のロールの完全な詳細を取得します。

```
/core-service=management/access=authorization/role-mapping= \
ROLENAME:read-resource(recursive=true)
{
    "outcome" => "success",
    "result" => {
        "include-all" => false,
        "exclude" => undefined,
        "include" => {
            "user-theboss" => {
                "name" => "theboss",
                "realm" => undefined,
                "type" => "USER"
            },
            "user-harold" => {
                "name" => "harold",
                "realm" => undefined,
                "type" => "USER"
            },
            "group-SysOps" => {
                "name" => "SysOps",
                "realm" => undefined,
                "type" => "GROUP"
            }
        }
    }
}
```

新規ロールの追加

この手順では、ロールの role-mapping エントリーを追加する方法を説明します。これは、ロールを設定する前に行う必要があります。

add 操作で、新しいロール設定を追加します。

```
/core-service=management/access=authorization/role-mapping=ROLENAME:add
```

グループに include されるユーザーの追加

この手順では、グループをロールの include されたリストに追加する方法を説明します。

ロールの設定が行われていない場合は、そのロールの role-mapping エントリーを最初に実行する必要があります。

add 操作を使用して、ロール、追加一覧にグループエントリーを追加します。

```
/core-service=management/access=authorization/role-mapping= \
ROLENAME/include=ALIAS:add(name=GROUPNAME, type=GROUP)
```

- ROLENAME は、設定されているロールの名前です。(例: 監査人)
- GROUPNAME は、investigators など、包含リストに追加されるグループの名前です。
- ALIAS は、このマッピングの一意の名前です。Red Hat は、group-GROUPNAME (例: group-investigators) などのエイリアスの命名規則を使用することを推奨します。

CLI の例

```
/core-service=management/access=authorization/role-mapping=Auditor/include= \
group-investigators:add(name=investigators, type=GROUP)
```

ロールに exclude されるグループの追加

この手順では、グループをロールの exclude されたリストに追加する方法を説明します。

ロールの設定が行われていない場合は、そのロールの role-mapping エントリーを最初に作成する必要があります。

add 操作を使用して、ロールの除外一覧にグループエントリーを追加します。

```
/core-service=management/access=authorization/role-mapping=ROLENAME/exclude= \
ALIAS:add(name=GROUPNAME, type=GROUP)
```

- ROLENAME は、設定されているロールの名前です (例: 監査人)
- GROUPNAME は、supervisors など、包含リストに追加されるグループの名前です。
- ALIAS は、このマッピングの一意の名前です。Red Hat は、group-GROUPNAME (例: group-supervisors) などのエイリアスの命名規則を使用することを推奨します。

CLI の例

```
/core-service=management/access=authorization/role-mapping=Auditor/exclude= \
group-supervisors:add(name=supervisors, type=GROUP)
```

グループーロールの include 設定の削除

この手順では、ロールマッピングからグループ包含エントリーを削除する方法を説明します。

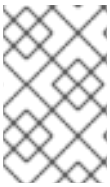
削除操作でエントリーを削除します。

```
/core-service=management/access=authorization/role-mapping=ROLENAME/include= \
ALIAS:remove
```

- ROLENAME は、設定されているロールの名前です (例: 監査人)
- ALIAS は、このマッピングの一意の名前です。Red Hat は、group-GROUPNAME (例: group-investigators) などのエイリアスの命名規則を使用することを推奨します。

CLI の例

```
/core-service=management/access=authorization/role-mapping=Auditor/include= \
group-investigators:remove
```



注記

包含の一覧からグループを削除しても、グループはシステムから削除されません。また、ロールがこのグループのユーザーに割り当てられないようにします。このロールは、引き続きグループのユーザーに割り当てられます。

ユーザーグループの exclude エントリーの削除

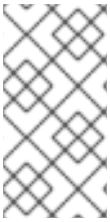
この手順では、ロールマッピングからグループ除外エントリーを削除する方法を説明します。

削除操作でエントリーを削除します。

```
/core-service=management/access=authorization/role-mapping=ROLENAME/exclude= \
ALIAS:remove
```

- ROLENAME は、設定されているロールの名前です。(例: 監査人)
- ALIAS は、このマッピングの一意の名前です。Red Hat は、group-GROUPNAME (例: group-supervisors) などのエイリアスの命名規則を使用することを推奨します。

```
/core-service=management/access=authorization/role-mapping=Auditor/exclude= \
group-supervisors:remove
```



注記

除外の一覧からグループを削除しても、システムからそのグループは削除されません。また、ロールがグループのメンバーに割り当てられることを保証する訳ではありません。依然としてロールはグループメンバーシップに基づいて除外される可能性があります。

3.3.6. LDAP での RBAC の使用

LDAP で RBAC を使用する基本と、JBoss EAP が LDAP で RBAC を使用するよう設定する方法は、JBoss EAP『How to Configure Identity Management Guide 』の「**LDAP and RBAC**」セクションを参照してください。

3.3.7. スコープ指定されたロール

スコープ指定されたロールは、標準的なロールのパーミッションを付与するユーザー定義のロールです。ただし、JBoss EAP 管理対象ドメインの1つ以上のサーバーグループまたはホストに対してのみ適用されます。スコープ指定されたロールでは、管理ユーザーに、必要なサーバーグループまたはホストのみに制限されるパーミッションを付与することができます。

重要

スコープ指定されたロールは、**Administrator** または **SuperUser** ロールが割り当てられているユーザーが作成できます。

これらは、以下の 5 つの特性によって定義されます。

- 一意名。
- ベースになる標準ロール。
- サーバーグループまたはホストへ適用されるかどうか。
- 制限されたサーバーグループまたはホストの一覧。
- すべてのユーザーが自動的に組み込まれるかどうか。デフォルトは false です。

作成すると、スコープ指定されたロールは標準ロールと同じ方法でユーザーおよびグループに割り当てることができます。

スコープ指定されたロールを作成しても、新しいパーミッションを定義することはできません。スコープ指定されたロールは、制限されたスコープ内で既存のロールのパーミッションを適用する場合にのみ使用できます。たとえば、スコープ指定されたロールは、単一のサーバーグループに制限されている **Deployer** ロールに基づいて作成できます。

ロールには、以下の 2 つのスコープのみを使用できます。

ホストスコープ指定ロール

ホストスコープ指定ロールは、そのロールのパーミッションは単一または複数のホストに制限されます。つまり、アクセスは関連する **/host=*/** リソースツリーに提供されますが、他のホストに固有のリソースは非表示になります。

サーバーグループスコープ指定ロール

サーバーグループスコープ指定ロールは、そのロールのパーミッションを 1 つ以上のサーバーグループに制限します。また、ロールのパーミッションは、指定した server-groups に関連付けられたプロファイル、ソケットバインディンググループ、サーバー設定、およびサーバーリソースにも適用されます。サーバーグループに論理的に関連していないものの内部のサブリソースは、ユーザーには認識できません。

3.3.7.1. 管理 CLI からのスコープ指定されたロールの設定

重要

この設定を実行できるのは、**SuperUser** または **Administrator** ロールのユーザーのみです。

新しいスコープ指定されたロールの追加

スコープ設定されたロールを新たに追加するには、以下の操作を行う必要があります。

```
/core-service=management/access=authorization/server-group-scoped-role= \
NEW-SCOPED-ROLE:add(base-role=BASE-ROLE, server-groups=[SERVER-GROUP-NAME])
```

```
/core-service=management/access=authorization/role-mapping=NEW-SCOPED-ROLE:add
```

NEW-SCOPED-ROLE、BASE-ROLE、および SERVER-GROUP-NAME を適切な情報に置き換えます。

スコープ指定ロールマッピングの表示および編集

スコープロールの詳細 (メンバーを含む) は、次のコマンドを発行することで表示できます。

```
/core-service=management/access=authorization/role-mapping= \
NEW-SCOPED-ROLE:read-resource(recursive=true)
```

NEW-SCOPED-ROLE を適切な情報に置き換えます。

スコープ指定ロールの詳細を編集するには、**write-attribute** コマンドを使用できます。以下に例を示します。

```
/core-service=management/access=authorization/role-mapping= \
NEW-SCOPED-ROLE:write-attribute(name=include-all, value=true)
```

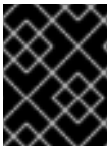
NEW-SCOPED-ROLE を適切な情報に置き換えます。

スコープ指定ロールの削除

```
/core-service=management/access=authorization/role-mapping= \
NEW-SCOPED-ROLE:remove
```

```
/core-service=management/access=authorization/server-group-scoped-role= \
NEW-SCOPED-ROLE:remove
```

NEW-SCOPED-ROLE を適切な情報に置き換えます。



重要

ユーザーまたはグループが割り当てられている場合、スコープ指定ロール は削除できません。最初にロールの割り当てを削除してから、スコープ指定ロールを削除します。

ユーザーの追加および削除

スコープ指定ロールへのユーザーの追加および削除は 標準ロールの追加と削除と同じプロセスに従います。

3.3.7.2. 管理コンソールからのスコープ指定ロールの設定

重要

この設定を実行できるのは、**SuperUser** または **Administrator** ロールのユーザーのみです。

管理コンソールのスコープ指定ロール設定は、以下の手順で確認できます。

1. 管理コンソールへログインします。
2. **管理** タブをクリックします
3. 左側の **アクセス制御** メニューで、**ロールの割り当て** を選択します。
4. **ロール** タブを選択してから、その中の **スコープロール** タブを選択します。

管理コンソールの **Scoped Roles** セクションは、現在設定されているスコープ指定ロールのリストが含まれるテーブルと、テーブルで現在選択されているロールの詳細を表示する Selection パネルの 2 つの主なエリアで構成されます。

スコープ指定ロールの設定タスクを実行する手順は次のとおりです。

新しいスコープ指定されたロールの追加

1. 管理コンソールへログインします。
2. **ロール** タブの **スコープロール** 領域にナビゲートします。
3. **Add** をクリックします。**スコープロールの追加** ダイアログが表示されます。
4. 以下の詳細を指定します。
   - **Name**: スコープ指定の新しいロールの一意の名前。
   - **Base Role**: このロールがパーミッションを元にするロール。
   - このロールをホストまたはサーバーグループのどちらに制限するかを **入力** します。
   - **スコープ**、ロールが制限されているホストまたはサーバーグループのリスト。複数のエントリーを選択できます。
   - このロールに **すべて** のユーザーが自動的に含まれる場合は、すべてを含めます。デフォルトは no です。
5. **保存** をクリックするとダイアログが閉じ、新しく作成されたロールがテーブルに表示されます。

スコープ指定されたロールの編集

1. 管理コンソールへログインします。
2. **ロール** タブの **スコープロール** 領域にナビゲートします。
3. テーブルで編集する目的のスコープロールをクリックします。そのロールの詳細は、表の下の **選択** パネルに表示されます。
4. **選択** パネルで **編集** をクリックします。**選択** パネルが編集モードになります。
5. 変更する詳細を更新して、**Save** ボタンをクリックします。**選択** パネルは前の状態に戻ります。**選択** パネルと表の両方に、新しく更新された詳細が表示されます。

スコープ指定されたロールメンバーの表示

1. 管理コンソールへログインします。
2. **ロール** タブの **スコープロール** 領域にナビゲートします。
3. テーブル内の目的のスコープロールをクリックしての **メンバー** を表示し、**メンバー** をクリックします。ロールの **メンバー** ダイアログが表示されます。ロールに含まれる、またはロールから除外されるユーザーとグループが表示されます。
4. この情報の確認が終了したら、**完了** をクリックします。

スコープ指定ロールの削除

1. 管理コンソールへログインします。

2. **ロール** タブの **スコープロール** 領域にナビゲートします。
3. テーブル上で、削除するスコープ指定ロールを選択します。
4. **Remove** ボタンをクリックします。**スコープロールの削除** ダイアログが表示されます。
5. **確認** をクリックします。ダイアログが閉じ、ロールが削除されます。

重要

ユーザーまたはグループが割り当てられている場合、スコープ指定ロール は削除できません。最初にロールの割り当てを削除してから、スコープ指定ロールを削除します。

ユーザーの追加および削除

スコープ指定ロールへのユーザーの追加および削除は 標準ロールの追加と削除と同じプロセスに従います。ユーザーのスコープ指定ロールを更新するには、以下を実行します。

1. 管理コンソールへログインします。
2. **管理** タブをクリックします
3. 左側の **アクセス制御** メニューで、**ロールの割り当て** を選択します。
4. **ユーザー** タブをクリックします。
5. テーブルから目的のユーザーを選択します。
6. **選択** セクションで **編集** をクリックします
7. 目的のロールを追加/削除したら、**保存** をクリックします。

3.3.8. 制約の設定

3.3.8.1. 感度制約の設定

各機密性制約は、機密であるとみなされるリソースのセットを定義します。通常、**機密**リソースとは、パスワードなどの秘密のリソースや、ネットワーキング、JVM 設定、システムプロパティーなどのサーバーに重大な影響を与えるリソースのことです。アクセス制御システム自体も機密であると見なされます。リソースの機密性は、どのロールが特定のリソースの読み取り、書き込み、またはアドレス指定できるかを制限します。

機密性制約設定は **/core-service=management/access=authorization/constraint=sensitivity-classification** にあります。

管理モデル内で、それぞれの機密性制約は分類として識別されます。分類はタイプにグループ化されます。13 種類に分類された 39 の分類が含まれています。

機密性制約を設定するには、**write-attribute** 操作を使用して **configured-requires-read**、**configured-requires-write**、**configured-requires-addressable** 属性を設定します。操作のタイプを機密に設定するには、属性の値を true に設定します。機密にしない場合は値を false に設定します。デフォルトでは、これらの属性は設定されず、default-requires-read、default-requires-write、default-requires-addressable の値が使用されます。設定した属性が適用されると、デフォルトではなく、その値が使用されます。デフォルト値は変更できません。

例: 読み取りシステムプロパティーを機密操作にする

```
/core-service=management/access=authorization/constraint= \
sensitivity-classification/type=core/classification= \
system-property:write-attribute(name=configured-requires-read, \
value=true)
```

結果

```
/core-service=management/access=authorization/constraint= \
sensitivity-classification/type=core/classification= \
system-property:read-resource
```

```
{
    "outcome" => "success",
    "result" => {
        "configured-requires-addressable" => undefined,
        "configured-requires-read" => true,
        "configured-requires-write" => undefined,
        "default-requires-addressable" => false,
        "default-requires-read" => false,
        "default-requires-write" => true,
        "applies-to" => {
            "/core-service=platform-mbean/type=runtime" => undefined,
            "/system-property=*" => undefined,
            "/" => undefined
        }
    }
}
```

これらの属性の設定に応じて、どのロールがどの操作を実行できるかを次の表に要約します。

表3.1 機密性制約設定の結果

値	requires-read	requires-write	requires-addressable
true	読み取りは機密です。Auditor、Administrator、SuperUser のみを読み取ることができます。	書き込みは機密です。Administrator および SuperUser のみが書き込み可能です。	アドレス指定は機密です。Auditor、Administrator、SuperUser のみがアドレス指定できます。
false	読み取りは機密ではありません。すべての管理ユーザーが読み取り可能です。	書き込みは機密ではありません。Maintainer、Administrator、および SuperUser のみが書き込み可能です。また、Deployer はリソースをアプリケーションリソースとして記述することもできます。	アドレス指定は機密ではありません。すべての管理ユーザーがアドレス指定できます。

3.3.8.2. アプリケーションリソース制約の設定

各アプリケーションリソース制約は、通常はアプリケーションとサービスのデプロイメントに関連するリソース、属性、および操作のセットを定義します。アプリケーションリソース制約が有効化されると、Deployer ロールの管理ユーザーに、適用されるリソースへのアクセスが付与されます。

アプリケーション制約設定は **/core-service=management/access=authorization/constraint=application-classification/** にあります。

各アプリケーションリソース制約は分類として識別されます。分類はタイプにグループ化されます。8 種類に分類された 14 の分類が含まれています。各分類には **apply-to** 要素があります。これは分類設定が適用されるパスパターンの一覧です。

デフォルトでは、有効になっている唯一のアプリケーションリソースの分類は コア です。コアには、デプロイメント、デプロイメントオーバーレイ、およびデプロイメント操作が含まれます。

アプリケーションリソースを有効にするには、**write-attribute** 操作を使用して、分類の **configured-application** 属性を **true** に設定します。アプリケーションリソースを無効にするには、この属性を false に設定します。デフォルトでは、これらの属性は設定されず、default-application 属性の値が使用されます。デフォルト値は変更できません。

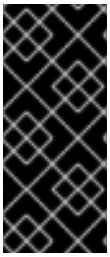
logger-profile アプリケーションリソースの分類を有効にする

```
/core-service=management/access=authorization/constraint= \
application-classification/type=logging/classification= \
logging-profile:write-attribute(name=configured-application, \
value=true)
```

結果

```
/core-service=management/access=authorization/constraint= \
application-classification/type=logging/classification= \
logging-profile:read-resource
```

```
{
    "outcome" => "success",
    "result" => {
        "configured-application" => true,
        "default-application" => false,
        "applies-to" => {"/subsystem=logging/logging-profile=*" => undefined}
    }
}
```



重要

アプリケーションリソース制約は、その設定に一致するすべてのリソースに適用されます。たとえば、**Deployer** ユーザーに、あるデータソースリソースへのアクセスを許可して、同じデータベースの別のリソースへのアクセスを拒否することはできません。このレベルの分離が必要な場合は、異なるサーバーグループでリソースを設定し、グループごとに異なるスコープ指定の Deployer ロールを作成することが推奨されます。

3.3.8.3. Vault 式制約の設定

デフォルトでは、vault 式の読み書きは機密操作です。Vault 式制約を設定すると、これらの操作のいずれかまたは両方を非機密に設定できます。この制約を変更すると、より多くのロールで vault 式の読み書きが可能になります。

Vault 式制約は **/core-service=management/access=authorization/constraint=vault-expression** にあります。

Vault 式制約を設定するには、**write-attribute** 操作を使用して **configured-requires-write** と **configured-requires-read** の値を **true** または **false** に設定します。デフォルトではそれらは設定されず、default-requires-read と default-requires-write の値が使用されます。デフォルト値は変更できません。

vault 式への書き込みを非機密操作にする

```
/core-service=management/access=authorization/constraint= \
vault-expression:write-attribute(name=configured-requires-write, \
value=false)
```

結果

```
/core-service=management/access=authorization/constraint= \
vault-expression:read-resource
```

```
{
    "outcome" => "success",
    "result" => {
        "configured-requires-read" => undefined,
        "configured-requires-write" => false,
        "default-requires-read" => true,
        "default-requires-write" => true
    }
}
```

この設定に応じて、どのロールがボールト式の読み取りと書き込みを実行できるかを次の表に要約します。

表3.2 vault 式制約の設定結果

値	requires-read	requires-write
true	読み取り操作は機密です。Auditor、Administrator、および SuperUser のみを読み取ることができます。	書き込み操作は機密です。Administrator および SuperUser のみが書き込み可能です。
false	読み取り操作は機密ではありません。すべての管理ユーザーは読み取りが可能です。	書き込み操作は機密ではありません。Monitor、Administrator、および SuperUser は書き込み可能です。Deployer は、vault 式がアプリケーションリソースにある場合にも書き込み可能です。

3.3.8.4. アプリケーションリソース制約に関する参考情報

タイプ: コア - 分類: デプロイメント - オーバーレイ

- デフォルト: true
- PATH: /deployment-overlay=*
- PATH: /deployment=*
- PATH: /
- upload-deployment-stream、full-replace-deployment、 upload-deployment-url、upload-deployment-bytes

タイプ: データソース - 分類: データソース

- デフォルト: false
- PATH: /deployment=*/subdeployment=*/subsystem=datasources/data-source=*
- パス: /subsystem=datasources/data-source=*
- /subsystem=datasources/data-source=ExampleDS
- PATH: /deployment=*/subsystem=datasources/data-source=*

タイプ: データソース - 分類:jdbc-driver

- デフォルト: false
- パス: /subsystem=datasources/jdbc-driver=*

タイプ: データソース - 分類:xa-data-source

- デフォルト: false
- パス: /subsystem=datasources/xa-data-source=*
- PATH: /deployment=*/subsystem=datasources/xa-data-source=*
- PATH: /deployment=*/subdeployment=*/subsystem=datasources/xa-data-source=*

タイプ: ロギング - 分類: ロガー

- デフォルト: false
- パス: /subsystem=logging/logger=*
- パス: /subsystem=logging/logging-profile=*/logger=*

タイプ: データソース - 分類:logging-profile

- デフォルト: false
- パス: /subsystem=logging/logging-profile=*

タイプ: メール - 分類: メールセッション

- デフォルト: false

- パス: /subsystem=mail/mail-session=*

タイプ: ネーミング - 分類: バインディング

- デフォルト: false
- パス: /subsystem=naming/binding=*

タイプ:resource-adapters- 分類:resource-adapters

- デフォルト: false
- パス: /subsystem=resource-adapters/resource-adapter=*

タイプ: セキュリティー - 分類: セキュリティードメイン

- デフォルト: false
- パス:/subsystem = security/security-domain = *

3.3.8.5. 機密性制約に関する参考情報

タイプ: コア - 分類: アクセス制御

- requires-addressable: true
- requires-read: true
- requires-write: true
- PATH: /core-service=management/access=authorization
- パス:/subsystem = jmx 属性:non-core-mbean-sensitive-sensitive

タイプ: コア - 分類: クレデンシャル

- requires-addressable: false
- requires-read: true
- requires-write: true
- パス:/subsystem = mail/mail-session = */server = pop3 属性: ユーザー名、パスワード
- パス:/subsystem = mail/mail-session = */server = imap 属性: ユーザー名、パスワード
- パス:/subsystem = datasources/xa-data-source = * 属性:user-name、recovery-username、password、recovery-password
- パス: /subsystem=mail/mail-session=*/custom=* ATTRIBUTE: username, password
- パス: /subsystem=datasources/data-source=*" ATTRIBUTE: user-name, password
- パス: /subsystem=remoting/remote-outbound-connection=*" ATTRIBUTE: username
- パス:/subsystem = mail/mail-session = */server = smtp 属性: ユーザー名、パスワード

- パス:/subsystem = web/connector = */configuration = ssl 属性: キーエイリアス、パスワード
- パス:/subsystem = resource-adapters/resource-adapter = */connection-definitions = * " 属性:recovery-username、 recovery-password

タイプ: コア - 分類: ドメインコントローラー

- requires-addressable: false
- requires-read: false
- requires-write: true

タイプ: コア - 分類: ドメイン名

- requires-addressable: false
- requires-read: false
- requires-write: true

タイプ: コア - 分類: 拡張

- requires-addressable: false
- requires-read: false
- requires-write: true
- PATH: /extension=*

タイプ: コア - 分類:jvm

- requires-addressable: false
- requires-read: false
- requires-write: true
- PATH: /core-service=platform-mbean/type=runtime ATTRIBUTE: input-arguments, boot-class-path, class-path, boot-class-path-supported, library-path

タイプ: コア - 分類: 管理 - インターフェイス

- requires-addressable: false
- requires-read: false
- requires-write: true
- /core-service=management/management-interface=native-interface
- /core-service=management/management-interface=http-interface

タイプ: コア - 分類: モジュールローディング

- requires-addressable: false
- requires-read: false
- requires-write: true
- PATH: /core-service=module-loading

タイプ: コア - 分類: パッチ

- requires-addressable: false
- requires-read: false
- requires-write: true
- PATH: /core-service=patching/addon=*
- PATH: /core-service=patching/layer=*"
- PATH: /core-service=patching

タイプ: コア - 分類:read-whole-config

- requires-addressable: false
- requires-read: true
- requires-write: true
- PATH: / OPERATION: read-config-as-xml

タイプ: コア - 分類: セキュリティードメイン

- requires-addressable: true
- requires-read: true
- requires-write: true
- パス:/subsystem = security/security-domain = *

タイプ: コア - 分類:security-domain-ref

- requires-addressable: true
- requires-read: true
- requires-write: true
- パス: /subsystem=datasources/xa-data-source=* ATTRIBUTE: security-domain
- パス: /subsystem=datasources/data-source=* ATTRIBUTE: security-domain
- パス:/subsystem = ejb3 属性:default-security-domain

- パス:/subsystem = resource-adapters/resource-adapter = */connection-definitions = * 属性:security-domain、recovery- security-domain、security-application、security-domain-and-application

タイプ: コア - 分類: セキュリティー - レルム

- requires-addressable: true
- requires-read: true
- requires-write: true
- PATH: /core-service=management/security-realm=*

タイプ: コア - 分類:security-realm-ref

- requires-addressable: true
- requires-read: true
- requires-write: true
- パス:/subsystem = remoting/connectedor = * 属性:security-realm
- PATH: /core-service=management/management-interface=native-interface ATTRIBUTE: security-realm
- PATH: /core-service=management/management-interface=http-interface ATTRIBUTE: security-realm
- パス:/subsystem = remoteing/remote-outbound-connection = * 属性:security-realm

タイプ: コア - 分類: セキュリティー - ボールト

- requires-addressable: false
- requires-read: false
- requires-write: true
- PATH: /core-service=vault

タイプ: コア - 分類: サービスコンテナー

- requires-addressable: false
- requires-read: false
- requires-write: true
- PATH: /core-service=service-container

タイプ: コア - 分類: スナップショット

- requires-addressable: false

- requires-read: false
- requires-write: false
- PATH: / ATTRIBUTE: take-snapshot, list-snapshots, delete-snapshot

タイプ: コア - 分類:socket-binding-ref

- requires-addressable: false
- requires-read: false
- requires-write: false
- パス:/subsystem = mail/mail-session = */server = pop3 属性: outbound-socket-binding-ref
- パス:/subsystem = mail/mail-session = */server = imap 属性: outbound-socket-binding-ref
- パス:/subsystem = remoting/connectedor = * 属性: ソケットバインディング
- パス: /subsystem=web/connector=* ATTRIBUTE: socket-binding
- パス:/subsystem = remoting/local-outbound-connection = * 属性: outbound-socket-binding-ref
- PATH: /socket-binding-group=*/local-destination-outbound-socket-binding=* ATTRIBUTE: socket-binding-ref
- パス:/subsystem = remoteing/remote-outbound-connection = * 属性: outbound-socket-binding-ref
- パス:/subsystem = mail/mail-session = */server = smtp 属性: outbound-socket-binding-ref
- パス:/subsystem = transaction 属性:process-id-socket-binding、status-socket-binding、socket-binding

タイプ: コア - 分類:socket-config

- requires-addressable: false
- requires-read: false
- requires-write: true
- PATH: /interface=* OPERATION: resolve-internet-address
- PATH: /core-service=management/management-interface=native-interface ATTRIBUTE: port, interface, socket-binding
- PATH: /socket-binding-group=*
- PATH: /core-service=management/management-interface=http-interface ATTRIBUTE: port, secure-port, interface, secure-socket-binding, socket-binding
- PATH: / OPERATION: resolve-internet-address
- パス:/subsystem = transactions 属性:process-id-socket-max-ports

タイプ: コア - 分類: システム - プロパティー

- requires-addressable: false
- requires-read: false
- requires-write: true
- PATH: /core-service=platform-mbean/type=runtime ATTRIBUTE: system-properties
- PATH: /system-property=*
- PATH: / OPERATION: resolve-expression

タイプ: データソース - 分類: データソース - セキュリティー

- requires-addressable: false
- requires-read: true
- requires-write: true
- パス:/subsystem = datasources/xa-data-source = * 属性: ユーザー名、セキュリティードメイン、パスワード
- パス:/subsystem = datasources/data-source = * 属性: ユーザー名、セキュリティードメイン、パスワード

タイプ:jdr- 分類:jdr

- requires-addressable: false
- requires-read: false
- requires-write: true
- パス:/subsystem = jdr 操作:generate-jdr-report

タイプ:jmx- 分類:jmx

- requires-addressable: false
- requires-read: false
- requires-write: true
- パス:/subsystem = jmx

タイプ: メール - 分類:mail-server-security

- requires-addressable: false
- requires-read: false
- requires-write: true
- パス:/subsystem = mail/mail-session = */server = pop3 属性:username、tls、ssl、password

- パス:/subsystem = mail/mail-session = */server = imap 属性: ユーザー名、tls、ssl、パスワード
- パス:/subsystem = mail/mail-session = **/custom =** 属性: ユーザー名、tls、ssl、パスワード
- パス:/subsystem = mail/mail-session = */server = smtp 属性: ユーザー名、tls、ssl、パスワード

タイプ: ネーミング - 分類:jndi-view

- requires-addressable: false
- requires-read: true
- requires-write: true
- パス:/subsystem =naming 操作:jndi-view

タイプ: ネーミング - 分類: ネーミング - バインディング

- requires-addressable: false
- requires-read: false
- requires-write: false
- パス: /subsystem=naming/binding=*

タイプ: リモーティング - 分類: リモーティング - セキュリティー

- requires-addressable: false
- requires-read: true
- requires-write: true
- パス:/subsystem = remoteing/connector = * 属性: 認証プロバイダー、セキュリティーレルム
- パス:/subsystem = remoteing/remote-outbound-connection = * 属性:username、security-realm
- パス: /subsystem=remoting/connector=*/security=sasl

タイプ:resource-adapters- 分類:resource-adapter-security

- requires-addressable: false
- requires-read: true
- requires-write: true
- パス:/subsystem = resource-adapters/resource-adapter = */connection-definitions = * 属性:security-domain、recovery-username、recovery-security-domain、security-application、security-domain-and-application、recovery- パスワード

タイプ: セキュリティー - 分類: その他 - セキュリティー

- requires-addressable: false

- requires-read: true
- requires-write: true
- パス:/subsystem = security 属性:deep-copy-subject-mode

タイプ:web- 分類:web-access-log

- requires-addressable: false
- requires-read: false
- requires-write: false
- パス: /subsystem=web/virtual-server=*/configuration=access-log

タイプ:web- 分類:web-connector

- requires-addressable: false
- requires-read: false
- requires-write: false
- パス: /subsystem=web/connector=*

タイプ:web- 分類:web-ssl

- requires-addressable: false
- requires-read: true
- requires-write: true
- パス: /subsystem=web/connector=*/configuration=ssl

タイプ:web- 分類:web-sso

- requires-addressable: false
- requires-read: true
- requires-write: true
- パス: /subsystem=web/virtual-server=*/configuration=sso

タイプ:web- 分類:web-valve

- requires-addressable: false
- requires-read: false
- requires-write: false
- パス: /subsystem=web/valve=*