Red Hat JBoss Enterprise Application Platform 7.0

サーバーセキュリティーの設定方法

Red Hat JBoss Enterprise Application Platform 7.0 向け

Last Updated: 2024-04-04

Red Hat JBoss Enterprise Application Platform 7.0 サーバーセキュリティーの設定方法

Red Hat JBoss Enterprise Application Platform 7.0 向け

法律上の通知

概要

本書の目的は、Red Hat JBoss Enterprise Application Platform 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。詳細は、JBoss EAP ですべての管理インターフェイスを保護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 7.0 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 インターフェイスを使用して設定の変更を行います。本書を完読することで、JBoss EAP の保護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 3

第2章 サーバーおよびインターフェイスの保護 4
2.1. ブロックの構築 4
2.2. 管理インターフェイスのセキュア化の方法 6
2.3. セキュリティー監査 26

第3章 管理対象ドメインのセキュア化 28
3.1. スレーブとドメインコントローラー間のパスワード認証の設定 28
3.2. ドメインとホストコントローラー間の SSL/TLS の設定 29

第4章 サーバーのユーザーと管理インターフェイスのセキュア化 33
4.1. ユーザー認証 33
4.2. 安全なパスワード 33
4.3. ロールベースのアクセス制御 48

第5章 JAVA SECURITY MANAGER 73
5.1. JAVA SECURITY MANAGER について 73
5.2. JAVA セキュリティーポリシーの定義 73
5.3. JAVA SECURITY MANAGER を使用した JBOSS EAP の実行 75
5.4. 以前のバージョンからの移行に関する考慮事項 76

第1章 セキュリティーの概要

JBoss EAP セキュリティーの基本と一般的なセキュリティー概念については、Red Hat JBoss Enterprise Application Platform セキュリティーアーキテクチャーガイド 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イスの保護

2.1. ブロックの構築

2.1.1. インターフェイスおよびソケットバインディング

JBoss EAP は、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に、ホストのインターフェイス (inet-address、nic など) とポートを利用します。これらのインターフェイスとポートは、JBoss EAP 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** 設定を通じて定義および設定されます。

インターフェイス と **socket-binding-group** を定義および設定する方法の詳細は、設定ガイドのソケットバインディングセクション を参照してください。

例: インターフェイス

```
<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>
</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="${jboss.socket.binding.port-offset:0}">
    <socket-binding name="management-http" interface="management"
port="${jboss.management.http.port:9990}"/>
    <socket-binding name="management-https" interface="management"
port="${jboss.management.https.port:9993}"/>
    <socket-binding name="ajp" port="${jboss.ajp.port:8009}"/>
    <socket-binding name="http" port="${jboss.http.port:8080}"/>
    <socket-binding name="https" port="${jboss.https.port:8443}"/>
    <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、管理インターフェイスで使用できる認証および承認メカニズム (ローカル、LDAP、プロパティーなど) を定義します。セキュリティーレルムの背景情報の詳細は、Red Hat JBoss Enterprise Application Platform セキュリティーアーキテクチャーガイド の **セキュリティーレルム** セクション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 では、既存のセキュリティーレルムの更新に加え、新しいセキュリティーレルムの作成も可能になります。管理コンソールで新しいセキュリティーレルムを作成したり、管理 CLI から以下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()
```

新しいセキュリティーレルムを作成し、認証または認可にプロパティーファイルを使用する場合は、新しいセキュリティードメイン専用の新しいプロパティーファイルを作成する必要があります。JBoss EAP は、他のセキュリティードメインによって使用される既存のファイルを再利用せず、設定ファイルが存在しない場合には、設定に指定された新しいファイルを自動的に作成します。

2.1.3. 管理インターフェイスのセキュア化にセキュリティーレルムとソケットバインディングを使用する

デフォルトでは、JBoss EAP は管理インターフェイスに接続するための http-interface を定義します。このインターフェイスは、JBoss EAP 設定の **<management-interfaces>** セクションで定義されます。

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

インターフェイスが **security-realm** と **socket-binding** 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セキュリティー保護できます。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セキュリティー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、セキュアなパスワード、JBoss EAP 内の他のサブシステムと重複する JMX など、JBoss EAP のセキュリティーに関連するトピックもいくつかあります。

2.2. 管理インターフェイスのセキュア化の方法

ここでは、JBoss EAP 管理インターフェイ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

ここで使用する管理 CLI コマンドは、JBoss EAP スタンドアロンサーバーを実行していることを仮定しています。JBoss EAP 管理対象ドメインの管理 CLI の使用の詳細は、JBoss EAP **管理 CLI ガイド** を参照してください。

2.2.1. JBoss EAP 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イ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 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は、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 の管理インターフェイスの設定

HTTPS を使用した通信のみを行うように JBoss EAP 管理インターフェイスを設定すると、セキュリティーが強化されます。クライアントと管理インターフェイス間のすべてのネットワークトラフィックは暗号化されるため、中間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、JBoss EAP インスタンスとの暗号化されていない通信を無効にします。この手順は、スタンドアロンサーバーとマネージドドメイン設定の両方に該当します。マネージドドメインの場合は、管理 CLI コマンドにホスト名 (例: /**host=master**) を付けます。

重要

管理インターフェイスで HTTPS を有効にする手順を実行している間は、明示的に指示されない限り、設定を再ロードしないでください。これを実行すると、管理インターフェイスがロックされる可能性があります。

1. キーストアを作成して、管理インターフェイスのセキュリティーを確保します。

注記

このキーストアは、管理インターフェイスが 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ターミナルからキーストアを生成します

-

```
$ 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. 管理インターフェイスが HTTPS にバインドされていることを確認します。

   a. スタンドアロンサーバーを実行しています。
   管理インターフェイスが HTTPS にバインドされるようにするには、**management-https** 設定を追加し、**management-http** 設定を削除する必要があります。

   以下の CLI コマンドを使用して管理インターフェイスを 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)
   ```

   b. マネージドドメインの起動
   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 socket 要素を変更します。

   以下のコマンドを使用して管理インターフェイスを HTTPS にバインドします。

   ```
   /host=master/core-service=management/management-interface=http-interface:write-
   attribute(name=secure-port,value=9993)

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

3. **オプション:** カスタム socket-binding-group を使用します。カスタムの **socket-binding-group** を使用する場合は、**management-https** バインディングが定義されていることを確認する必要があります。デフォルトではポート **9993** にバインドされます。これを確認するには、サーバーの設定ファイルの **socket-binding-group** セクションを確認するか、管理 CLI を使用します。

   管理 CLI を使用して socket-binding-group 設定を読み取る例

   ```
   /socket-binding-group=standard-sockets/socket-binding=management-https:read-
   resource(recursive=true)

   {
       "outcome" => "success",
       "result" => {
           "client-mappings" => undefined,
           "fixed-port" => false,
           "interface" => "management",
           "multicast-address" => undefined,
           "multicast-port" => undefined,
           "name" => "management-https",
           "port" => expression "${jboss.management.https.port:9993}"
       }
   }
   ```

4. 新しいセキュリティーレルムを作成します。この例では、HTTPS を使用する新しいセキュリティーレルム ManagementRealmHTTPS は、**EAP_HOME/standalone/configuration/** ディレクトリーにある **https-mgmt-users.properties** という名前の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します。ユーザー名とパスワードは後でファイルに追加できます。ただし、現在は、**https-mgmt-users.properties** という名前で空のファイルを作成し、その場所に保存する必要があります。以下の例は、**touch** コマンドの使用を示しています。しかし、テキストエディターなどの他のメカニズムを使用することもできます。

   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
   ```

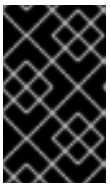
   次に、次の CLI コマンドを入力して、ManagementRealmHTTPS という名前の新しいセキュリティーレルムを作成します。

   ```
   /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)
   ```

   これで、新しいセキュリティーレルムを作成し、認証用にプロパティーファイルを使用するよう設定しました。次に、**EAP_HOME/bin/** ディレクトリーの **add-user** スクリプトを使用して、そのプロパティーファイルにユーザーを追加する必要があります。**add-user** スクリプトを実行する際には、**-up** と **-r** オプションをそれぞれ使用してプロパティーファイルとセキュリティーレルムの両方を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ユーザー名とパスワード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ー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。

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します。

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

6. キーストアを使用するように管理インターフェイスを設定します。
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インターフェイスを設定します。パラメーターファイル、パスワード、およびエイリアスについては、その値を 最初の手順 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)
```

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)
```

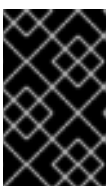
このときは、サーバーの設定をリロードする必要があります。

```
reload
```

サーバー設定のリロード後には、起動したサービスの数を示すテキストの直前に以下の内容がログに含まれるはずです。

```
13:50:54,160 INFO  [org.jboss.as] (Controller Boot Thread) WFLYSRV0061: Http management interface listening on https://127.0.0.1:9993/management
13:50:54,162 INFO  [org.jboss.as] (Controller Boot Thread) WFLYSRV0052: Admin console listening on https://127.0.0.1:9993
```

これで、管理インターフェイスはポート **9993** 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。



重要

このとき、CLI は接続を切断し、ポートバインディングが変更されたため再接続できなくなります。次の手順 に進み、**jboss-cli.xml** を更新して CLI が再接続できるようにします。

7. **jboss-cli.xml** を更新します。
管理 CLI を使用して管理アクションを実行する場合は、**EAP_HOME/bin/jboss-cli.xml** ファイルに次の変更を加える必要があります。

- **<default-protocol>** の値を **https-remoting** に更新します。
- **<default-controller>** において、**<protocol>** の値を **https-remoting** に更新します。
- **<default-controller>** で、**<port>** の値を **9993** に更新します。

jboss-cli.xml の例

```
<jboss-cli xmlns="urn:jboss:cli:2.0">
    <default-protocol use-legacy-override="true">https-remoting</default-protocol>
    <!-- The default controller to connect to when 'connect' command is executed w/o
arguments -->
    <default-controller>
        <protocol>https-remoting</protocol>
        <host>localhost</host>
        <port>9993</port>
    </default-controller>
...
```

次回、管理 CLI を使用して管理インターフェイスに接続する場合は、サーバー証明書を受け入れ、**ManagementRealmHTTPS** セキュリティーレルムに対して認証を行う必要があります。

サーバー証明書の受け入れと認証の例

```
$ ./jboss-cli.sh -c
Unable to connect due to unrecognised server certificate
Subject    - CN=appserver,OU=Sales,O=Systems Inc,L=Raleigh,ST=NC,C=US
Issuer     - CN=appserver, OU=Sales, O=Systems Inc, L=Raleigh, ST=NC, C=US
Valid From - Tue Jun 28 13:38:48 CDT 2016
Valid To   - Thu Jun 28 13:38:48 CDT 2018
MD5 : 76:f4:81:8b:7e:c3:be:6d:ee:63:c1:7a:b7:b8:f0:fb
SHA1 : ea:e3:f1:eb:53:90:69:d0:c9:69:4a:5a:a3:20:8f:76:c1:e6:66:b6

Accept certificate? [N]o, [T]emporarily, [P]ermenantly : p
Authenticating against security realm: ManagementRealmHTTPS
Username: httpUser
Password:
[standalone@localhost:9993 /]
```

2.2.3. 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 インターフェイ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)
```

2.2.4. 管理インターフェイスの双方向 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証明書があるという点で、HTTPS 用の管理インターフェイスの設定 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。たとえば、**cn=jboss,dc=redhat,dc=com** となります。

CA_HOST2
: HOST2 証明書に使用する DN (識別名) です。例: **cn=myclient,dc=redhat,dc=com**

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 7 セキュリティーアーキテクチャーガイド の パスワード Vault セクションおよび **パスワード Vault システム** セクションを参照してください。

警告

Red Hat では、影響するすべてのパッケージで TLSv1.1 または TLSv1.2 を利用するために SSLv2、SSLv3、および TLSv1.0 を明示的に無効化することを推奨しています。

1. キー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. 対立する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。
サーバー (**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イスを指定します。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. **http-interface** の **security-realm** を新しい CertificateRealm に変更します。

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

6. CLI の SSL/TLS 設定を追加します。

重要

双方向 SSL/TLS の追加に加え、管理インターフェイスも HTTPS にバインドされるように設定 してください。

EAP_HOME/bin/jboss-cli.xml を設定ファイルとして使用する CLI の SSL/TLS 設定を追加します。

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、変数に適切な値を使用して SSL/TLS 設定を追加します。

jboss-cli.xml XML 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>
```

パスワード vault に保存されているキーストアおよびトラストストアパスワードを使用するには、vault 設定および適切な vault の値を **EAP_HOME/bin/jboss-cli.xml** に追加する必要があります。

jboss-cli.xml XML 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>
```

2.2.5. アプリケーション用の SSL/TLS の設定

JBoss EAP では、管理インターフェイスの HTTPS および双方向 SSL/TLS のサポートに加えて、セキュリティードメインで使用するために SSL/TLS (HTTPS リスナー経由) を設定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 (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 キーと証明書の生成例については、管理インターフェイスの双方向 SSL/TLS の設定セクション の最初の 2 つの手順を参照してください。HTTPS リスナー (暗号スイートを含む) の詳細は、HTTPS リスナーのリファレンスセクション を参照してください。

一方向 SSL/TLS の設定
この例では、キーストアの **identity.jks** がサーバー設定ディレクトリーにコピーされ、指定のプロパティーで設定されたことを仮定します。管理者は、example の独自の値を置き換えてください。

注記

ここで使用する管理 CLI コマンドは、JBoss EAP スタンドアロンサーバーを実行していることを仮定しています。JBoss EAP 管理対象ドメインの管理 CLI の使用の詳細は、JBoss EAP **管理 CLI ガイド** を参照してください。

1. 最初に HTTPS セキュリティーレルムを追加し、設定します。HTTPS セキュリティーレルムが設定されたら、セキュリティーレルムを参照する **undertow** サブシステムで **https-listener** を設定します。

```
batch

/core-service=management/security-realm=HTTPSRealm/:add

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

/subsystem=undertow/server=default-server/https-listener=https:add( \
socket-binding=https, security-realm=HTTPSRealm)

run-batch
```

警告

Red Hat では、影響するすべてのパッケージで TLSv1.1 または TLSv1.2 を利用するために SSLv2、SSLv3、および TLSv1.0 を明示的に無効化することを推奨しています。

2. 変更を反映するためにサーバーをリロードします。

```
reload
```

2.2.6. アプリケーションの双方向 SSL/TLS の設定

アプリケーションの双方向 SSL/TLS の設定は 管理インターフェイスの双方向 SSL/TLS の設定 で概説されているのと同じ手順の多くに従います。アプリケーションに双方向 SSL/TLS を設定するには、次の手順を実行する必要があります。

1. クライアントとサーバー両方のストアを生成します。
2. クライアントとサーバー両方の証明書をエクスポートします。
3. opposing トラストストアに証明書をインポートします。
4. サーバーのキーストアおよびトラストストアを使用するサーバーで、**CertificateRealm** などのセキュリティーレルムを定義します。
5. セキュリティーレルムを使用し、クライアント検証が必須となるように **undertow** サブシステムを更新します。

最初の 4 つの手順については 管理インターフェイスの双方向 SSL/TLS の設定 で説明されています。

重要

新しいセキュリティーレルムが追加されても、サーバーがリロードされていない場合は、サーバーをリロードしてから次の手順を実行する必要があります。

Undertow サブシステムの調整

キーストア、証明書、トラストストア、およびセキュリティーレルムが作成され、設定されると、**undertow** サブシステムに HTTPS リスナーを追加し、作成したセキュリティーレルムを使用する必要があり、クライアントの検証が必須になります。

```
/subsystem=undertow/server=default-server/https-listener=https:add( \
socket-binding=https, security-realm=CertificateRealm, verify-client=REQUIRED)
```

重要

アプリケーションに双方向 SSL/TLS を有効化した JBoss EAP インスタンスに接続するクライアントは、クライアント証明書またはキーストアにアクセスできなければなりません。つまり、クライアントのキーストアに証明書が含まれるクライアントキーストアが必要になります。クライアントがブラウザーを使用して JBoss EAP インスタンスに接続している場合は、その証明書またはキーストアをブラウザーの証明書マネージャーにインポートする必要があります。

注記

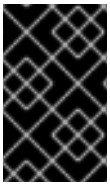
アプリケーションでの双方向 SSL/TLS に加えて、アプリケーションで証明書ベースの認証を使用する方法の詳細は、JBoss EAP の **アイデンティティー Management の設定方法** の 証明書ベースの認証を使用するためのセキュリティードメインの設定 を参照してください。

2.2.7. HTTPS リスナーのリファレンス

HTTPS リスナーで使用できる属性の完全なリストについては、JBoss EAP **設定ガイド** の Undertow サブシステム属性 を参照してください。

2.2.7.1. 暗号化スイートについて

許可される一連の暗号を設定できます。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 JVM のデフォルトです。



重要

強度の低い暗号を使用すると、セキュリティー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/8/docs/technotes/guides/security/StandardNames.html#Cipher を参照してください。

有効な暗号スイートのリストを更新するには、**undertow** サブシステムの HTTPS リスナーの **enabled-cipher-suites** 属性を使用します。

CLI の例

```
/subsystem=undertow/server=default-server/https-listener=https:write-attribute(name=enabled-cipher-
suites,value="TLS_RSA_WITH_AES_128_CBC_SHA,TLS_RSA_WITH_AES_256_CBC_SHA")
```

注記

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。

2.2.8. Red Hat Enterprise Linux 6 での SSL/TLS の FIPS 140-2 暗号の有効化

Undertow は、SSL/TLS に FIPS 140-2 準拠の暗号を使用するように設定できます。この設定例の範囲は、FIPS モードで Mozilla NSS ライブラリーを使用する Red Hat Enterprise Linux 6 に限定されます。

重要

Red Hat Enterprise Linux 6 は、FIPS140-2 に準拠するように設定済みである。詳細は、https://access.redhat.com/knowledge/solutions/137833 を参照してください。

警告

TLS 1.2 プロトコルは、FIPS モードで実行されている場合、Oracle/OpenJDK および JBoss EAP ではサポートされていないため、**NoSuchAlgorithmException** が発生する可能性があります。この問題の詳細は、こちら を参照してください。

重要

SSL/TLS の FIPS 140-2 準拠暗号化が有効になっている環境で **jboss-cli.sh** を実行すると、**FIPS mode: only SunJSSE TrustManagers may be used** というエラーが表示されます。**jboss-cli.sh** ファイルの **javax.net.ssl.keyStore** および **javax.net.ssl.trustStore** システムプロパティーを更新することで、この問題を回避できます。

```
JAVA_OPTS="$JAVA_OPTS -Djavax.net.ssl.trustStore=NONE -
Djavax.net.ssl.trustStoreType=PKCS11"
JAVA_OPTS="$JAVA_OPTS -Djavax.net.ssl.keyStore=NONE -
Djavax.net.ssl.keyStoreType=PKCS11 -
Djavax.net.ssl.keyStorePassword=imapassword"
```

SSL/TLS に FIPS 140-2 に準拠する暗号を使用するように Undertow を設定するには、以下を行う必要があります。

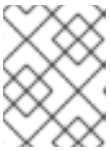
- NSS データベースの設定
- Undertow の設定

2.2.8.1. NSS データベースの設定

1. NSS データベースを格納するために、適切なユーザーが所有するディレクトリーを作成します。

   NSS データベースディレクトリーを作成するコマンドの例

   ```
   $ mkdir -p  /usr/share/jboss-as/nssdb
   $ chown jboss /usr/share/jboss-as/nssdb
   $ modutil -create -dbdir /usr/share/jboss-as/nssdb
   ```

   注記

   jboss ユーザーはあくまで例です。実際には、JBoss EAP の実行に使用するオペレーティングシステム上のユーザーに置き換える必要があります。

2. NSS 設定ファイル (**/usr/share/jboss-as/nss_pkcsll_fips.cfg**) を作成します。
   以下を指定する必要があります。

   - 名前
   - NSS ライブラリーが置かれているディレクトリー
   - 以前のステップで NSS データベースが作成されたディレクトリー

   例 nss_pkcsll_fips.cfg

   ```
   name = nss-fips
   nssLibraryDirectory=/usr/lib64
   nssSecmodDirectory=/usr/share/jboss-as/nssdb
   nssDbMode = readOnly
   nssModule = fips
   ```

注記

64 ビットバージョンの Red Hat Enterprise Linux 6 使用していない場合は、**nssLibraryDirectory** を **/usr/lib64** ではなく、**/usr/lib** に置き換えます。

3. **$JAVA_HOME/jre/lib/security/java.security** 設定ファイルを編集します。
次の行を **$JAVA_HOME/jre/lib/security/java.security** に追加します。

例 java.security

```
security.provider.1=sun.security.pkcs11.SunPKCS11  /usr/share/jboss-as/nss_pkcsll_fips.cfg
```

注記

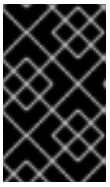
上記の行で指定されている **nss_pkcsll_fips.cfg** 設定ファイルは、前のステップで作成したファイルです。

$JAVA_HOME/jre/lib/security/java.security の次のリンクも更新する必要があります。

```
security.provider.5=com.sun.net.ssl.internal.ssl.Provider
```

to

```
security.provider.5=com.sun.net.ssl.internal.ssl.Provider  SunPKCS11-nss-fips
```



重要

このファイルの他の **security.provider.X** 行 (例: **security.provider.2**) には、このプロバイダーに優先順位が指定されるように X の値を増やす必要があります。

4. 前の手順で作成した NSS データベースディレクトリーで **modutil** コマンドを実行して、FIPS モードを有効化します。

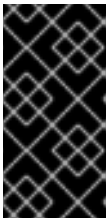
```
modutil -fips true -dbdir /usr/share/jboss-as/nssdb
```

注記

この時点では、NSS 共有オブジェクトの一部に対するライブラリー署名の再生成を求めるセキュリティーライブラリーエラーが発生することがあります。

5. FIPS トークンにパスワードを設定します。
トークンの名前は、**NSS FIPS 140-2 Certificate DB**である **必要があります** 。

```
modutil -changepw "NSS FIPS 140-2 Certificate DB" -dbdir /usr/share/jboss-as/nssdb
```



重要

FIPS トークンに使用されるパスワードは、FIPS に準拠したパスワードである必要があります。パスワードの強度が不十分な場合は、以下のようなエラーが発生することがあります。**ERROR: Unable to change password on token "NSS FIPS 140-2 Certificate DB"**

6. NSS ツールを使用して証明書を作成します。

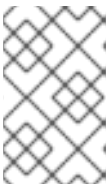
コマンド例

```
certutil -S -k rsa -n undertow  -t "u,u,u" -x -s "CN=localhost, OU=MYOU, O=MYORG, L=MYCITY, ST=MYSTATE, C=MY" -d /usr/share/jboss-as/nssdb
```

2.2.8.2. Undertow の設定

SSL/TLS の FIPS 140-2 準拠暗号化のセットアップを完了するには、次の手順を実行します。

1. SSL/TLS を使用するように Undertow を設定します。



注記

以下のコマンドはバッチモードで実行する必要があります。あるいは、**ssl** サーバーアイデンティティーを追加した後にサーバーをリロードする必要があります。以下の例はバッチモードを使用しています。

```
batch

/core-service=management/security-realm=HTTPSRealm:add

/core-service=management/security-realm=HTTPSRealm/server-identity=ssl:add(keystore-provider=PKCS11, keystore-password="strongP@ssword1")

/subsystem=undertow/server=default-server/https-listener=https:add(socket-binding=https, security-realm=HTTPSRealm, enabled-protocols="TLSv1.1")

run-batch
```

Undertow を SSL / TLS に設定する基本的な詳細は、Setting up an SSL/TLS for Applications で説明されています。

2. Undertow によって使用される暗号スイートを設定します。
SSL/TLS を設定した後は、特定のセットの暗号スイートを有効にするために https リスナーとセキュリティーレルムを設定する必要があります。

必要な暗号スイート

```
SSL_RSA_WITH_3DES_EDE_CBC_SHA, SSL_DHE_RSA_WITH_3DES_EDE_CBC_SHA,
TLS_RSA_WITH_AES_128_CBC_SHA, TLS_DHE_DSS_WITH_AES_128_CBC_SHA,
TLS_DHE_RSA_WITH_AES_128_CBC_SHA, TLS_RSA_WITH_AES_256_CBC_SHA,
TLS_DHE_DSS_WITH_AES_256_CBC_SHA,
TLS_DHE_RSA_WITH_AES_256_CBC_SHA,
TLS_ECDH_ECDSA_WITH_3DES_EDE_CBC_SHA,
```

```
TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA,
TLS_ECDH_ECDSA_WITH_AES_256_CBC_SHA,
TLS_ECDHE_ECDSA_WITH_3DES_EDE_CBC_SHA,
TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA,
TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA,
TLS_ECDH_RSA_WITH_3DES_EDE_CBC_SHA,
TLS_ECDH_RSA_WITH_AES_128_CBC_SHA,
TLS_ECDH_RSA_WITH_AES_256_CBC_SHA,
TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA,
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA,
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA,
TLS_ECDH_anon_WITH_3DES_EDE_CBC_SHA,
TLS_ECDH_anon_WITH_AES_128_CBC_SHA,
TLS_ECDH_anon_WITH_AES_256_CBC_SHA
```

https リスナーに暗号スイートを有効にする基本的な方法は、暗号スイートについて で説明されています。https リスナーで暗号化スイートを有効にするには、以下を行います。

HTTPS リスナーでの暗号スイートを有効にするコマンドの例

```
/subsystem=undertow/server=default-server/https-listener=https:write-
attribute(name=enabled-cipher-
suites,value="SSL_RSA_WITH_3DES_EDE_CBC_SHA,SSL_DHE_RSA_WITH_3DES_EDE_
CBC_SHA,TLS_RSA_WITH_AES_128_CBC_SHA,TLS_DHE_DSS_WITH_AES_128_CBC_S
HA,TLS_DHE_RSA_WITH_AES_128_CBC_SHA,TLS_RSA_WITH_AES_256_CBC_SHA,TLS
_DHE_DSS_WITH_AES_256_CBC_SHA,TLS_DHE_RSA_WITH_AES_256_CBC_SHA,TLS_
ECDH_ECDSA_WITH_3DES_EDE_CBC_SHA,TLS_ECDH_ECDSA_WITH_AES_128_CBC_
SHA,TLS_ECDH_ECDSA_WITH_AES_256_CBC_SHA,TLS_ECDHE_ECDSA_WITH_3DES_
EDE_CBC_SHA,TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA,TLS_ECDHE_ECDSA_
WITH_AES_256_CBC_SHA,TLS_ECDH_RSA_WITH_3DES_EDE_CBC_SHA,TLS_ECDH_R
SA_WITH_AES_128_CBC_SHA,TLS_ECDH_RSA_WITH_AES_256_CBC_SHA,TLS_ECDHE
_RSA_WITH_3DES_EDE_CBC_SHA,TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA,TLS_
ECDHE_RSA_WITH_AES_256_CBC_SHA,TLS_ECDH_anon_WITH_3DES_EDE_CBC_SHA
,TLS_ECDH_anon_WITH_AES_128_CBC_SHA,TLS_ECDH_anon_WITH_AES_256_CBC_S
HA")
```

3. セキュリティーレルムで暗号スイートを有効にします。

セキュリティーレルムで暗号スイートを有効化するコマンドの例

```
/core-service=management/security-realm=HTTPSRealm/server-identity=ssl:write-
attribute(name=enabled-cipher-suites, value=[SSL_RSA_WITH_3DES_EDE_CBC_SHA,
SSL_DHE_RSA_WITH_3DES_EDE_CBC_SHA, TLS_RSA_WITH_AES_128_CBC_SHA,
TLS_DHE_DSS_WITH_AES_128_CBC_SHA,
TLS_DHE_RSA_WITH_AES_128_CBC_SHA, TLS_RSA_WITH_AES_256_CBC_SHA,
TLS_DHE_DSS_WITH_AES_256_CBC_SHA,
TLS_DHE_RSA_WITH_AES_256_CBC_SHA,
TLS_ECDH_ECDSA_WITH_3DES_EDE_CBC_SHA,
TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA,
TLS_ECDH_ECDSA_WITH_AES_256_CBC_SHA,
TLS_ECDHE_ECDSA_WITH_3DES_EDE_CBC_SHA,
TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA,
TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA,
TLS_ECDH_RSA_WITH_3DES_EDE_CBC_SHA,
TLS_ECDH_RSA_WITH_AES_128_CBC_SHA,
```

```
TLS_ECDH_RSA_WITH_AES_256_CBC_SHA,
TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA,
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA,
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA,
TLS_ECDH_anon_WITH_3DES_EDE_CBC_SHA,
TLS_ECDH_anon_WITH_AES_128_CBC_SHA,
TLS_ECDH_anon_WITH_AES_256_CBC_SHA])
```

4. 以下のコマンドを実行することで、JVM が PKCS11 キーストアから秘密鍵を読み取りできることを確認します。

```
keytool -list -storetype pkcs11
```

2.2.9. IBM JDK における FIPS 140-2 準拠暗号化

IBM JDK では、マルチプラットフォーム用の IBM JCE (Java Cryptographic Extension) IBMJCEFIPS プロバイダーと IBM JSSE (Java Secure Sockets Extension) FIPS 140-2 暗号化モジュール (IBMJSSE2) が FIPS 140-2 準拠の暗号化を提供します。

IBMJCEFIPS プロバイダーの詳細は、IBM JCEFIPS の IBM ドキュメント および NIST IBMJCEFIPS – セキュリティーポリシー を参照してください。IBMJSSE2 の詳細は、 ここ を参照してください。

2.2.9.1. キーストレージ

IBM JCE はキーストアを提供しません。このキーはコンピューターに保存され、その物理境界は残しません。このキーがコンピューター間で移動している場合は暗号化する必要があります。

FIPS 準拠モードで **keytool** を実行するには、次のように各コマンドで **-providerClass** オプションを使用します。

```
keytool -list -storetype JCEKS -keystore mystore.jck -storepass mystorepass -providerClass
com.ibm.crypto.fips.provider.IBMJCEFIPS
```

2.2.9.2. FIPS プロバイダー情報の確認

サーバーによって使用される IBMJCEFIPS に関する情報を調べるには、**-Djavax.net.debug=true** を**standalone.conf** または **domain.conf** に追加してデバッグレベルのロギングを有効にします。FIPS プロバイダーに関する情報は、**server.log** に記録されます。次に例を示します。

```
04:22:45,685 INFO  [stdout] (http-/127.0.0.1:8443-1) JsseJCE:  Using MessageDigest SHA from
provider IBMJCEFIPS version 1.7
04:22:45,689 INFO  [stdout] (http-/127.0.0.1:8443-1) DHCrypt:  DH KeyPairGenerator  from provider
from init IBMJCEFIPS version 1.7
04:22:45,754 INFO  [stdout] (http-/127.0.0.1:8443-1) JsseJCE:  Using KeyFactory DiffieHellman from
provider IBMJCEFIPS version 1.7
04:22:45,754 INFO  [stdout] (http-/127.0.0.1:8443-1) JsseJCE:  Using KeyAgreement DiffieHellman
from provider IBMJCEFIPS version 1.7
04:22:45,754 INFO  [stdout] (http-/127.0.0.1:8443-1) DHCrypt:  DH KeyAgreement  from provider
IBMJCEFIPS version 1.7
04:22:45,754 INFO  [stdout] (http-/127.0.0.1:8443-1) DHCrypt:  DH KeyAgreement  from provider
from initIBMJCEFIPS version 1.7
```

2.2.10. JVM が FIPS モードで実行されているときにマネージドドメインを起動する

重要

管理対象ドメインがあり、FIPS が設定されており、必要な証明書がすべて設定されていることを前提としています。これには、ドメインコントローラーの証明書を各コントローラーのトラストストアにインポートすることが含まれます。管理対象ドメインの設定の詳細は、JBoss EAP 設定ガイドの ドメイン管理 を参照してください。FIPS の設定の詳細は、Red Hat Enterprise Linux 6 での SSL/TLS の FIPS 140-2 暗号化の有効化 を参照してください。

通信に SSL/TLS を使用するには、各ホストコントローラーとマスタードメインコントローラーを更新する必要があります。

警告

Red Hat では、影響するすべてのパッケージで TLSv1.1 を利用するために SSLv2、SSLv3、および TLSv1.0 を明示的に無効化することを推奨しています。

1. マスタードメインコントローラー上に SSL/TLS セキュリティーレルムを作成します。
NSS データベースを PKCS11 プロバイダーとして使用するように設定されたマスタードメインコントローラー上に SSL/TLS セキュリティーレルムを作成する必要があります。

セキュリティーレルムの例

```
<security-realm name="HTTPSRealm">
    <server-identities>
        <ssl>
            <engine enabled-protocols="TLSv1.1"/>
            <keystore provider="PKCS11" keystore-password="strongP@ssword1"/>
        </ssl>
    </server-identities>
    <authentication>
        <local default-user="\$local"/>
        <properties path="https-users.properties" relative-to="jboss.domain.config.dir"/>
    </authentication>
</security-realm>
```

2. 各ホストコントローラーに SSL/TLS セキュリティーレルムを作成します。
認証用に SSL/TLS トラストストアを使用してセキュリティーレルムを作成する必要があります。

セキュリティーレルムの例

```
<security-realm name="HTTPSRealm">
  <authentication>
    <truststore provider="PKCS11" keystore-password="strongP@ssword1"/>
  </authentication>
</security-realm>
```

注記

各ホストでこのプロセスを繰り返す必要があります。

3. マスタードメインコントローラー上のネイティブインターフェイスを保護します。
マスタードメインコントローラーのネイティブインターフェイスが、作成したばかりのセキュリティーレルムで保護されていることを確認する必要があります。

ネイティブインターフェイスの例

```
<management-interfaces>
  ...
  <native-interface security-realm="HTTPSRealm">
    <socket interface="management" port="${jboss.management.native.port:9999}"/>
  </native-interface>
</management-interfaces>
```

4. 各ホストコントローラーで SSL/TLS レルムを使用して、マスタードメインコントローラーに接続します。
マスタードメインコントローラーへの接続に使用されるセキュリティー領域を更新する必要があります。この変更は、サーバーが動作していないときにホストコントローラーの設定ファイル (**host.xml**、**host-slave.xml** など) で直接行う必要があります。

ホストコントローラー設定ファイルの例

```
<domain-controller>
  <remote security-realm="HTTPSRealm">
    <discovery-options>
      <static-discovery name="primary" protocol="${jboss.domain.master.protocol:remote}"
host="${jboss.domain.master.address}" port="${jboss.domain.master.port:9999}"/>
    </discovery-options>
  </remote>
</domain-controller>
```

5. 各サーバーがホストコントローラーに接続する方法を更新します。
また、各サーバーがホストコントローラーに接続する方法も更新する必要があります。

サーバー設定の例

```
<server name="my-server" group="my-server-group">
  <ssl ssl-protocol="TLS" trust-manager-algorithm="SunX509" truststore-type="PKCS11"
truststore-password="strongP@ssword1"/>
</server>
```

6. マネージドドメインで双方向 SSL/TLS を設定します。
双方向 SSL/TLS を有効にするには、マスタードメインコントローラーの SSL/TLS セキュリティーレルムにトラストストア認証メソッドを追加します。以下の管理 CLI コマンドを実行します。

```
/host=master/core-service=management/security-
realm=HTTPSRealm/authentication=truststore:add(keystore-provider="PKCS11",keystore-
password="strongP@ssword1")

reload --host=master
```

SSL サーバーアイデンティティーを持つように各ホストコントローラーのセキュリティーレルムを更新する必要もあります。以下の管理 CLI コマンドを実行します。

```
/host=host1/core-service=management/security-realm=HTTPSRealm/server-identity=ssl:add(keystore-provider=PKCS11, keystore-password="strongP@ssword1",enabled-protocols=["TLSv1.1"])

reload --host=host1
```

重要

また、各ホストの証明書がドメインコントローラーのトラストストアにインポートされることを確認する必要があります。

2.2.11. 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管理操作がリモートでトリガーできます。JBoss EAP で JMX へのリモートアクセスを無効にするには、**jmx** サブシステムのリモーティングコネクターを削除します。

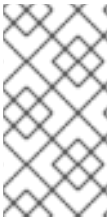
リモーティングコネクターの削除

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

JMX 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ガイドの JMX セクション を参照してください。

2.2.12. 管理インターフェイ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 および宣言型セキュリティーに関する詳細と背景については、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宣言型セキュリティーと JAAS セクション を参照してください。



注記

JBoss EAP 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イスのセキュア化はサポートされません。**ADMIN_ONLY** モードの詳細は、JBoss EAP **設定ガイド** の ADMIN_ONLY モードでの JBoss EAP の実行 セクションを参照してください。

JAAS を使用して管理インターフェイスに認証するには、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。
   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。

```
/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 **http-interface** 管理インターフェイスを更新します。

```
/core-service=management/management-interface=http-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、RBAC (Role-Based Access Control) に使用されます。

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

2.2.13. サイレント認証

JBoss EAP 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。この機能は便宜上、ローカルユーザーが認証を必要とせずに管理 CLI スクリプトを実行できるよう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
```

2.2.14. Undertow 応答ヘッダーの削除

デフォルトの JBoss EAP **undertow** サブシステムには、**default-host** によって各 HTTP 応答に追加される 2 つの応答ヘッダーが含まれています。

- **Server**。**JBoss-EAP/7** に設定されています
- **X-Powered-By**。**Undertow/1** に設定されています

これらは開発やデバッグの目的には役立ちますが、使用中のサーバーに関する情報を公開しない場合は、これらのヘッダーを削除することを検討してください。

次の管理 CLI コマンドを使用して、これらの応答ヘッダーを **default-host** から削除します。

```
/subsystem=undertow/server=default-server/host=default-host/filter-ref=server-header:remove

/subsystem=undertow/server=default-server/host=default-host/filter-ref=x-powered-by-header:remove

reload
```

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するイベントに応じて、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. セキュリティードメインのセキュリティー監査の設定

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
   - 画面上部の **Configuration** をクリックします。
   - マネージドドメインで、左上の **Profile** 選択ボックスから変更するプロファイルを選択します。
   - **Subsystems** をクリックし、**Security** をクリックします。
   - 編集するセキュリティードメインをクリックし、**View** をクリックします。
2. 監査設定に移動します。
   画面左側の **Audit** をクリックします。

   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。
   Add をクリックして、**Code** 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。また、**Name** セクションに任意の名前を入力します。
4. モジュールが動作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監視する手段を提供することです。この監視は、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 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、監査ログファイルを確認します。

5. **オプション:** モジュールオプションを追加、編集、または削除します。
モジュールにオプションを追加するには、Modules リストでそのエントリーをクリックし、ページの Details セクションで Module Options タブを選択します。Add 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、Remove をクリックして削除し、Add 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 管理対象ドメインのセキュア化

管理インターフェイスをセキュア化する他に、マネージドドメインの JBoss EAP インスタンス間の通信をセキュアにすることもできます。

管理対象ドメインの動作モードの概念と一般的な設定については、JBoss EAP **設定ガイド** の ドメイン管理 セクションを参照してください。

3.1. スレーブとドメインコントローラー間のパスワード認証の設定

マネージドドメインを設定する場合は、デフォルトでマスタードメインコントローラーが、接続する各スレーブコントローラーに認証を必要とするよう設定されます。スレーブのコントローラーを適切な認証情報で設定するには、以下を行う必要があります。

1. マスタードメインコントローラーにユーザーを追加します。
   add-user スクリプトを使用して、マスタードメインコントローラーにユーザーを追加する必要があります。具体的には、マスターが管理インターフェイスを保護するために使用するのと同じレルム (デフォルトでは **ManagementRealm**) にユーザーが追加されていることを確認する必要があります。また、**Is this new user going to be used for one AS process to connect to another AS process?** という質問に必ず **yes** と回答する必要があります。

重要

ユーザーを追加すると、スクリプトは次のステップで使用される **<secret>** 要素を出力します。この値は、次の手順で使用するために維持する必要があります。

スレーブユーザーの追加例

```
$ EAP_HOME/bin/add-user.sh

What type of user do you wish to add?
 a) Management User (mgmt-users.properties)
 b) Application User (application-users.properties)
(a): a

Enter the details of the new user to add.
Using realm 'ManagementRealm' as discovered from the existing property files.
Username : slave-user
Password recommendations are listed below. To modify these restrictions edit the add-user.properties configuration file.
 - The password should be different from the username
 - The password should not be one of the following restricted values {root, admin, administrator}
 - The password should contain at least 8 characters, 1 alphabetic character(s), 1 digit(s), 1 non-alphanumeric symbol(s)
Password :
Re-enter Password :
What groups do you want this user to belong to? (Please enter a comma separated list, or leave blank for none)[  ]:
About to add user 'slave-user' for realm 'ManagementRealm'
Is this correct yes/no? yes
Added user 'slave-user' to file '/home/user/EAP-7.0.0/jboss-eap-7.0/standalone/configuration/mgmt-users.properties'
Added user 'slave-user' to file '/home/user/EAP-7.0.0/jboss-eap-
```

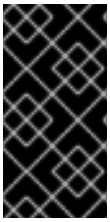
```
7.0/domain/configuration/mgmt-users.properties'
Added user 'slave-user' with groups  to file '/home/user/EAP-7.0.0/jboss-eap-
7.0/standalone/configuration/mgmt-groups.properties'
Added user 'slave-user' with groups  to file '/home/user/EAP-7.0.0/jboss-eap-
7.0/domain/configuration/mgmt-groups.properties'
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? yes
To represent the user add the following to the server-identities definition <secret
value="ABCzc3dv11Qx" />
```

2. クレデンシャルを使用するようにスレーブコントローラーを設定します。
マスタードメインコントローラーでユーザーを作成したら、ホスト設定ファイル (例: **host.xml** または **host-slave.xml**) で、その認証ファイルを使用するように各スレーブコントローラーを更新する必要があります。これを行うには、**domain controller** 設定の **<remote>** 要素にユーザー名を追加する必要があります。また、**<remote>** 要素を保護するために使用されるレルムの **server identities** に **<secret>** を追加する必要があります。直前の手順でユーザーをマスタードメインコントローラーに追加する際に、ユーザー名と **<secret>** の両方が取得されていました。

例

```
...
<security-realm name="ManagementRealm">
    <server-identities>
        <!-- Replace this with either a base64 password of your own, or use a vault with a vault
expression -->
        <secret value="ABCzc3dv11Qx"/>
    </server-identities>
...
<domain-controller>
  <remote security-realm="ManagementRealm" username="slave-user">
      <discovery-options>
          <static-discovery name="primary" protocol="${jboss.domain.master.protocol:remote}"
host="${jboss.domain.master.address}" port="${jboss.domain.master.port:9999}"/>
      </discovery-options>
  </remote>
</domain-controller>
```

3.2. ドメインとホストコントローラー間の SSL/TLS の設定



重要

マネージドドメインの JBoss EAP インスタンス間で使用する SSL/TLS を設定する場合は、対話に応じて各インスタンスにクライアントまたはサーバーのロールを設定できます。これには、すべてのホストコントローラーとドメインコントローラーが含まれます。そのため、エンドポイント間で双方向 SSL/TLS を設定することを推奨します。

マスタードメインコントローラーとホストコントローラー間で相互に通信する際に SSL / TLS を使用するようにマネージドドメインの JBoss EAP インスタンスを設定できます。そのためには、次の手順を実行する必要があります。

1. 必要なすべての証明書およびキーストアを生成して設定します。

エンドポイント間で双方向 SSL/TLS を設定するには、マスタードメインコントローラーおよび各ホストコントローラーの証明書およびキーストアを生成し、設定する必要があります。マスタードメインコントローラーの証明書を各ホストコントローラーのキーストアにインポートし、各ホストコントローラーの証明書をマスタードメインコントローラーのキーストアにインポートする必要があります。このプロセスの詳細は 管理インターフェイスの双方向 SSL/TLS の設定 で説明されています。

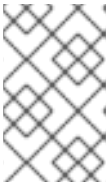
2. マスタードメインコントローラーを、SSL/TLS を使用するように設定します。
すべての証明書とキーストアを設定したら、双方向 SSL/TLS を使用するようにセキュリティーレルムを設定する必要があります。これは、SSL/TLS を使用して、これを認証に必須するようにセキュリティーレリムを設定することで実行されます。その後、そのセキュリティーレルムはホストコントローラーとマスタードメインコントローラー間の接続に使用される管理インターフェイスのセキュア化に使用されます。

注記

以下のコマンドはバッチモードで実行する必要があります。あるいは、**ssl** サーバーアイデンティティーを追加した後にサーバーをリロードする必要があります。以下の例はバッチモードを使用しています。

```
batch

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

/host=master/core-service=management/security-realm=CertificateRealm/server-identity=ssl:add(alias=domaincontroller,keystore-relative-to=jboss.domain.config.dir,keystore-path=domaincontroller.jks,keystore-password=secret)

/host=master/core-service=management/security-realm=CertificateRealm/authentication=truststore:add(keystore-relative-to=jboss.domain.config.dir,keystore-path=domaincontroller.jks,keystore-password=secret)

/host=master/core-service=management/security-realm=CertificateRealm/authentication=local:add(default-user=\$local)

/host=master/core-service=management/security-realm=CertificateRealm/authentication=properties:add(relative-to=jboss.domain.config.dir,path=mgmt-users.properties)

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

run-batch
```

3. SSL/TLS を使用するようすべてのホストコントローラーを設定します。
双方向 SSL/TLS を使用するようマスタードメインコントローラーを設定した場合は、各ホストコントローラーも使用するように設定する必要があります。このプロセスはマスタードメインコントローラーの設定とほぼ同じです。ただし、各ホストに固有のキーストアを使用する必要があります。



注記

以下のコマンドはバッチモードで実行する必要があります。あるいは、**ssl** サーバーアイデンティティーを追加した後にサーバーをリロードする必要があります。以下の例はバッチモードを使用しています。

```
batch

/host=instance1/core-service=management/security-realm=CertificateRealm:add()

/host=instance1/core-service=management/security-realm=CertificateRealm/server-identity=ssl:add(alias=instance1,keystore-relative-to=jboss.domain.config.dir,keystore-path=instance1.jks,keystore-password=secret)

/host=instance1/core-service=management/security-realm=CertificateRealm/authentication=truststore:add(keystore-relative-to=jboss.domain.config.dir,keystore-path=instance1.jks,keystore-password=secret)

/host=instance1/core-service=management/security-realm=CertificateRealm/authentication=local:add(default-user="\$local")

/host=instance1/core-service=management/security-realm=CertificateRealm/authentication=properties:add(relative-to=jboss.domain.config.dir,path=mgmt-users.properties)

/host=instance1/core-service=management/management-interface=native-interface:write-attribute(name=security-realm,value=CertificateRealm)

run-batch
```

また、マスタードメインコントローラーの接続時に使用するセキュリティーレルムを更新する必要があります。この変更は、サーバーが実行されていないときに、ホストコントローラーの設定ファイル (**host.xml** または **host-slave.xml** など) で直接行う必要があります。

ホストコントローラー設定ファイルの例

```
<domain-controller>
        <remote security-realm="CertificateRealm" username="slave-user">
            <discovery-options>
                <static-discovery name="primary"
protocol="${jboss.domain.master.protocol:remote}" host="${jboss.domain.master.address}"
port="${jboss.domain.master.port:9999}"/>
            </discovery-options>
        </remote>
    </domain-controller>
```

警告

Red Hat では、影響するすべてのパッケージで TLSv1.1 または TLSv1.2 を利用するために SSLv2、SSLv3、および TLSv1.0 を明示的に無効化することを推奨しています。

第4章 サーバーのユーザーと管理インターフェイスのセキュア化

JBoss EAP のさまざまなインターフェイスのセキュリティーを確保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。

4.1. ユーザー認証

4.1.1. 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イ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イスとリモートインターフェイスという 2 つの方法で管理インターフェイスにアクセスできます。これら両方の認証メカニズムの基本については 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 デフォルトセキュリティー および Red Hat JBoss Enterprise Application Platform Out of the Box 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イスへのアクセスは**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イスが有効になり、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。

4.1.2. 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イスのセキュア化にも対応しています。LDAP の基礎と JBoss EAP での LDAP の動作方法については、Red Hat JBoss Enterprise Application Platform 7 セキュリティーアーキテクチャーガイド の LDAP、管理インターフェイスでの LDAP の使用、および ManagementRealm での LDAP の使用 のセクション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イスを保護する方法の詳細は、アイデンティティー管理の設定方法ガイド の **LDAP を使用した管理インターフェイスのセキュリティー保護** セクションを参照してください。

4.2. 安全なパスワード

4.2.1. 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 つで設定されます。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。

重要

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。

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。

4.2.1.1. パスワード vault の設定

以下の手順に従って、パスワード Vault をセットアップし、使用します。

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 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。

4.2.1.2. パスワード Vault を初期化します。

パスワード vault は、各パラメーターの値の入力を求めるプロンプトが表示される場合にインタラクティブに初期化できます。またはコマンドラインにすべてのパラメーター値が提供される場合に非対話的に初期化されます。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。キーストアと同じ場所に暗号化された情報をすべて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 が実行されるユーザーアカウントでは、最低でも読み書きアクセスが必要です。キーストアは、パスワード vault の設定 時に作成したディレクトリーに置く必要があります。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。

password vault コマンドを非対話的に実行するには、関連情報のパラメーターを指定して **vault** スクリプト (**EAP_HOME/bin/** にあります) を呼び出すことができます。

```
$ 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

=========================================================================

Nov 09, 2015 9:02:47 PM org.picketbox.plugins.vault.PicketBoxSecurityVault init
INFO: PBOX00361: Default Security Vault Implementation Initialized and Ready
WFLYSEC0047: 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
********************************************
WFLYSEC0048: Vault Configuration in WildFly 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> ...
************************************************
```

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。
   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.batt** 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
Nov 09, 2015 9:24:36 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。

4.2.1.3. パスワード vault を使用するように JBoss EAP を設定します。

パスワードおよびその他の機密属性をマスクして設定ファイルで使用する前に、JBoss EAP 7 はそれらを保存および復号化するパスワード vault を認識する必要があります。

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

```
/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")])

/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。

4.2.2. 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。セキュリティー上の理由からも、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。

password vault コマンドを非対話的に実行するには、関連情報のパラメーターを指定して **vault** スクリプト (**EAP_HOME/bin/** にあります) を呼び出すことができます。

```
$ 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
```

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

```
$ 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

=========================================================================

Nov 09, 2015 9:24:36 PM org.picketbox.plugins.vault.PicketBoxSecurityVault init
INFO: PBOX00361: Default Security Vault Implementation Initialized and Ready
WFLYSEC0047: 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
********************************************
WFLYSEC0048: Vault Configuration in WildFly configuration file:
********************************************
...
</extensions>
<vault>
  <vault-option name="KEYSTORE_URL" value="../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="../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
```

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。
   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イスを起動し、**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
 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
Nov 09, 2015 9:24:36 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
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
```

4.2.2.1. 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>
...
```

4.2.2.2. 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"
)*/
```

4.2.2.3. 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。

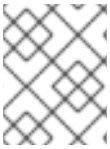
パスワード vault に機密文字列を保存または使用する前に、その文字列がすでに保存されているかどうかを確認すると便利です。

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。

非対話形式で、すべてのパラメーター値を一括で提供します。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。password vault コマンドを非対話的に実行するには、関連情報のパラメーターを指定して **vault** スクリプト (**EAP_HOME/bin/** にあります) を呼び出すことができます。

```
$ 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。

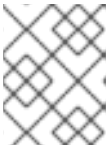
```
Password already exists.
```

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.
```

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。
   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.batt** 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
Nov 09, 2015 9:24:36 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
```

4.2.2.4. 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

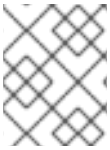
前提条件として、パスワード vault から機密文字列を削除するには、JBoss EAP の設定で使用されるかどうかを確認します。

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。

非対話形式で、すべてのパラメーター値を一括で提供します。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。password vault コマンドを非対話的に実行するには、関連情報のパラメーターを指定して **vault** スクリプト (**EAP_HOME/bin/** にあります) を呼び出すことができます。

```
$ 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, 2015 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。
   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.batt** 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、セキュリティー保護された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
```

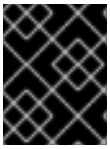
出力例

```
 *********************************
 ****  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
```

4.2.2.5. 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てください。詳細は、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 にカスタム実装を使用するには、以下を実行します。

1. インターフェイス **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、インターフェイスが **SecurityVault** である **org.picketbox** で依存関係を指定します。
3. 次の属性を持つ 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")])
```

4.2.2.6. 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}...

'...' は正確なコマンドを指します。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 2 つ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** 出力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リストです。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。

4.3. ロールベースのアクセス制御

ロールベースアクセス制御の基本については、**セキュリティーアーキテクチャーガイド** の ロールベースアクセス制御 および 管理インターフェイスへの RBAC の追加 のセクションで説明されています。

4.3.1. 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、管理 CLI はサーバーと同じマシン上で実行される場合、**SuperUser** ロールとして実行されます。

管理 CLI を使用して RBAC を有効にするには、アクセス承認リソースの **write-attribute** 操作を使用して、**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 を使用して RBAC を無効にするには、アクセス承認リソースの **write-attribute** 操作を使用して、**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>
```

4.3.2. 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設定します。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>
```

4.3.3. 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 7 は、ユーザーとグループの両方のメンバーシップに基づいて包含と除外のシステムを使用し、ユーザーがどのロールに属するかを決定します。

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リスト表示される、または
- ロールに含まれるようにリスト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リスト、または
- ロールから除外されるリスト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。

4.3.3.1. 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: Auditor)。

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/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: Auditor)
- 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 (例: max)。

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: Auditor)
- 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: Auditor)
- 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

包含のリスト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

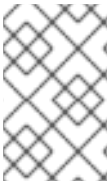
この手順では、ロールマッピングからユーザー除外エントリーを削除する方法を説明します。

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: Auditor)
- 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

除外のリスト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。

4.3.4. 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。

4.3.5. 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、ロールの包含リスト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: Auditor)
- GROUPNAME は、包含リスト (例: investigators) 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)
```

ロールを除外グループとして追加

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、ロールの除外リスト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: Auditor)
- GROUPNAME は、包含リスト (例: supervisors) 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包含設定の削除

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。

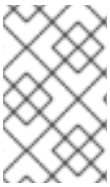
削除操作でエントリーを削除します。

```
/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: Auditor)
- 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

包含のリスト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: Auditor)
- 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

除外のリスト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。

4.3.6. LDAP での RBAC の使用

LDAP で RBAC を使用する基本と、LDAP で RBAC を使用するように JBoss EAP 7 を設定する方法は、JBoss EAP アイデンティティー管理の設定方法 の **LDAP および RBAC** セクションを参照してください。

4.3.7. 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 マネージドドメインの 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リスト。
- 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

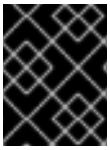
ホストスコープ指定ロールは、そのロールのパーミッションは単一または複数のホストに制限されます。つまり、アクセスは関連する **/host=*/** リソースツリーに提供されますが、他のホストに固有のリソースは非表示になります。

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。

4.3.7.1. 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/role-mapping=NEW-SCOPED-ROLE:add
```

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

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

スコープ指定されたロールの詳細 (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。

4.3.7.2. 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. 管理コンソールへのログイン
2. Access Control タブをクリックします
3. 左側の Roles メニューをクリックすると、すべてのロール (スコープ指定されたロールを含む) が表示されます。

以下の手順では、スコープ指定ロールの設定タスクを実行する方法について説明します。

新しいスコープ指定されたロールの追加

- 管理コンソールへのログイン
- Access Control タブをクリックします
- 左側の Roles メニューをクリックします。
- Add をクリックします。
- 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 Type します。
  - Scope、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 Include All 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。
- Save 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

- 管理コンソールへのログイン
- **Access Control** タブをクリックします
- 左側の **Roles** メニューをクリックします。
- 編集するスコープ付きロールをクリックし、**Edit** をクリックします。
- 変更する詳細を更新して、**Save** ボタンをクリックします。

スコープ指定されたロールメンバーの表示

- 管理コンソールへのログイン
- **Access Control** タブをクリックします
- 左側の **Roles** メニューをクリックします。
- 目的のスコープロールをクリックし、**Include** または **Exclude** を選択して、含めるメンバーまたは除外するメンバーを表示します。

スコープ指定ロールの削除

- 管理コンソールへのログイン
- **Access Control** タブをクリックします
- 左側の **Roles** メニューをクリックします。
- 目的のスコープロールをクリックし、**Edit** ボタンの横にあるドロップダウン矢印をクリックして、**Remove** をクリックします。
- **Confirm** 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。

- 管理コンソールへのログイン
- **Access Control** タブをクリックします
- 左側の **Roles** メニューをクリックします。
- 目的のスコープロールをクリックし、**Include** または **Exclude** を選択して、含めるメンバーまたは除外するメンバーを表示します。
- メンバーを追加するには、**Add** をクリックし、含めるメンバーまたは除外するメンバーを選択して、**Save** をクリックします。

- メンバーを削除するには、削除するメンバーを選択し、**Remove** をクリックします。

4.3.8. 制約の設定

4.3.8.1. 機密性制約の設定

各機密性制約は、**sensitive** 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テーブルに要約します。

表4.1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。

4.3.8.2. 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リソースパスパターンのリストです。

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

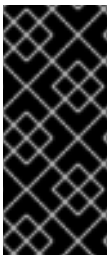
```
/core-service=management/access=authorization/constraint=application-classification/type=logging/classification=logging-profile:write-attribute(name=configured-application,value=true)
```

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。

4.3.8.3. 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** の属性を **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 vault 式の読み取りと書き込みを実行できるかを次のテーブルに要約します。

表4.2 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。

4.3.8.4. 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: /
- 操作: 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: mail-session

- 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: security-domain

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

4.3.8.5. 機密性制約に関する参考情報

タイプ: コア - 分類: access-control

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

タイプ: コア - 分類: クレデンシャル

- requires-addressable: false
- requires-read: true
- requires-write: true
- PATH: /subsystem=mail/mail-session=*/server=pop3 ATTRIBUTE: username , password
- PATH: /subsystem=mail/mail-session=*/server=imap ATTRIBUTE: username , password
- PATH: /subsystem=datasources/xa-data-source=* ATTRIBUTE: 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
- PATH: /subsystem=mail/mail-session=*/server=smtp ATTRIBUTE: username, password
- PATH: /subsystem=resource-adapters/resource-adapter=*/connection-definitions=*" ATTRIBUTE: recovery-username, recovery-password

タイプ: コア - 分類: domain-controller

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

タイプ: コア - 分類: domain-names

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

タイプ: コア - 分類: エクステンション

- 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: management-interfaces

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

タイプ: コア - 分類: module-loading

- 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: security-domain

- 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: security-realm

- 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
- PATH: /subsystem=remoting/connector=* ATTRIBUTE: 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=remoting/remote-outbound-connection=* 属性: security-realm

タイプ: コア - 分類: security-vault

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

タイプ: コア - 分類: service-container

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

タイプ: コア - 分類: snapshots

- 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
- PATH: /subsystem=mail/mail-session=*/server=pop3 ATTRIBUTE: outbound-socket-binding-ref
- PATH: /subsystem=mail/mail-session=*/server=imap ATTRIBUTE: outbound-socket-binding-ref
- PATH: /subsystem=remoting/connector=* ATTRIBUTE: socket-binding
- PATH: /subsystem=remoting/local-outbound-connection=* ATTRIBUTE: outbound-socket-binding-ref
- PATH: /socket-binding-group=*/local-destination-outbound-socket-binding=* ATTRIBUTE: socket-binding-ref
- PATH: /subsystem=remoting/remote-outbound-connection=* ATTRIBUTE: outbound-socket-binding-ref
- PATH: /subsystem=mail/mail-session=*/server=smtp ATTRIBUTE: outbound-socket-binding-ref
- パス:/ /subsystem=transactions 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: /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: system-property

- 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: data-source-security

- requires-addressable: false
- requires-read: true
- requires-write: true
- PATH: /subsystem=datasources/xa-data-source=* ATTRIBUTE: user-name, security-domain, password
- パス: /subsystem=datasources/data-source=* 属性: user-name、security-domain、password

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
- PATH: /subsystem=mail/mail-session=*/server=pop3 ATTRIBUTE: username, tls, ssl, password
- パス: /subsystem=mail/mail-session=*/server=imap 属性: ユーザー名、tls、ssl、password
- パス: /subsystem=mail/mail-session=**/custom=** 属性: username、tls、ssl、password
- パス: /subsystem=mail/mail-session=*/server=smtp 属性: username、tls、ssl、password

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: naming-binding

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

タイプ: remoting - 分類: remoting-security

- requires-addressable: false
- requires-read: true
- requires-write: true
- PATH: /subsystem=remoting/connector=* ATTRIBUTE: authentication-provider, security-realm
- PATH: /subsystem=remoting/remote-outbound-connection=* ATTRIBUTE: 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-password

タイプ: セキュリティー - 分類: misc-security

- requires-addressable: false
- requires-read: true
- requires-write: true
- PATH: /subsystem=security ATTRIBUTE: deep-copy-subject-mode

第5章 JAVA SECURITY MANAGER

5.1. JAVA SECURITY MANAGER について

Java Security Manager は、Java Virtual Machine (JVM) サンドボックスの外部境界を管理するクラスで、JVM 内で実行されるコードが JVM 外のリソースと対話する方法を制御します。Java Security Manager を有効にすると、Java API はさまざまな潜在的に危険な操作を実行する前に、セキュリティーマネージャーで承認を確認します。Java Security Manager はセキュリティーポリシーを使用して、特定のアクションが許可または拒否されるかどうかを判断します。

5.2. JAVA セキュリティーポリシーの定義

Java セキュリティーポリシーは、さまざまなクラスのコードに対する定義されたパーミッションのセットです。Java Security Manager は、アプリケーションによって要求されたアクションをセキュリティーポリシーと比較します。ポリシーがアクションを許可した場合、Security Manager はそのアクションの実行を許可します。ポリシーによりアクションが許可されない場合、セキュリティーマネージャーはそのアクションを拒否します。

重要

以前のバージョンの JBoss EAP では、外部ファイル (例: **EAP_HOME/bin/server.policy**) を使用してポリシーが定義されていました。JBoss EAP 7 では、**security-manager** サブシステムと、個別のデプロイメントで XML ファイルを使用するという 2 つの方法で Java セキュリティーポリシーを定義します。**Security-manager** サブシステムは、**ALL** ディプロイメントの最小および最大パーミッションを定義します。一方、XML ファイルは、個別のディプロイメントによって要求されたパーミッションを指定します。

5.2.1. Security Manager サブシステムでのポリシーの定義

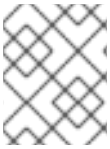
security-manager サブシステムを使用すると、すべてのデプロイメントの共有のパーミッションまたは共通のパーミッションを定義できます。これは、最小および最大のパーミッションセットを定義することで実行できます。すべてのデプロイメントには、最小パーミッションで定義された少なくともすべてのパーミッションが付与されます。このデプロイメントプロセスは、最大パーミッションセットで定義されたものを超えるパーミッションを要求すると失敗します。

最小権限を更新するコマンド例

```
/subsystem=security-manager/deployment-permissions=default:write-attribute(name=minimum-permissions, value=[{class="java.util.PropertyPermission", actions="read", name="*"}])
```

最大権限を更新するコマンド例

```
/subsystem=security-manager/deployment-permissions=default:write-attribute(name=maximum-permissions, value=[{class="java.util.PropertyPermission", actions="read,write", name="*"},
{class="java.io.FilePermission", actions="read,write", name="/-"}])
```



注記

最大パーミッションセットが定義されていない場合、その値のデフォルトが **java.security.AllPermission** に設定されます。

security-manager サブシステムの完全リファレンスは、JBoss EAP**設定ガイド** にあります。

5.2.2. デプロイメントでのポリシーの定義

JBoss EAP 7 では、JSR 342 の一部であり、Java EE 7 仕様の一部である **META-INF/permissions.xml** をデプロイメントに追加できます。このファイルでは、デプロイメントに必要なパーミッションを指定できます。最低限のパーミッションセットが **security-manager** サブシステムに定義され、**META-INF/permissions.xml** がデプロイメントに追加されると、これらのパーミッションの結合が許可されます。**permissions.xml** で要求されるパーミッションが **security-manager** サブシステムで定義された最大ポリシーを超える場合、そのデプロイメントは成功しません。**META-INF/permissions.xml** と **META-INF/jboss-permissions.xml** の両方がデプロイメントにある場合は、**META-INF/jboss-permissions.xml** で要求されるパーミッションのみが付与されます。

Java EE 7 仕様は **permission.xml** がアプリケーション全体またはトップレベルのデプロイメントモジュールに対応するように指定します。サブデプロイメントに特定のパーミッションを定義する必要がある場合は、JBoss EAP 固有の **META-INF/jboss-permissions.xml** を使用できます。これは、**permission.xml** と同じ形式に従います。また、宣言されるデプロイメントモジュールにのみ適用されます。

サンプルの permissions.xml

```
<permissions version="7">
  <permission>
  <class-name>java.util.PropertyPermission</class-name>
  <name>*</name>
  <actions>read</actions>
  </permission>
</permissions>
```

5.2.3. モジュールにおけるポリシーの定義

<permissions> 要素を **module.xml** ファイルに追加してモジュールのパーミッションを制限できます。**<permissions>** 要素には、モジュールに付与するパーミッションを定義する、ゼロまたはそれ以上の **<grant>** 要素が含まれます。**<grant>** 要素には以下の属性が含まれます。

permission
付与するパーミッションの修飾クラス名。

name
パーミッションクラスのコンストラクターに提供するパーミッション名。

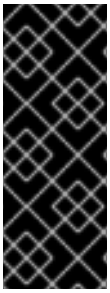
actions
一部のパーミッションタイプで必要なアクションの (オプション) リスト。

ポリシーが定義された module.xml の例

```
<module xmlns="urn:jboss:module:1.5" name="org.jboss.test.example">
  <permissions>
    <grant permission="java.util.PropertyPermission" name="*" actions="read,write" />
    <grant permission="java.io.FilePermission" name="/etc/-" actions="read" />
  </permissions>
  ...
</module>
```

<permissions> 要素が存在する場合、モジュールはリストしたパーミッションのみに制限されます。**<permissions>** 要素が存在しない場合は、モジュールには制限がつきません。

5.3. JAVA SECURITY MANAGER を使用した JBOSS EAP の実行



重要

以前のバージョンの JBoss EAP では、**-Djava.security.manager** Java システムプロパティーとカスタムセキュリティーマネージャーの使用が許可されていました。どちらも JBoss EAP 7 ではサポートされていません。さらに、Java セキュリティーマネージャーポリシーは、**security-manager** サブシステム内で定義されるようになりました。つまり、外部ポリシーファイルと **-Djava.security.policy** Java システムプロパティーは JBoss EAP 7 ではサポートされません。

重要

Java Security Manager を有効にして JBoss EAP を起動する前に、すべてのセキュリティーポリシーが **security-manager** サブシステムで定義されていることを確認する必要があります。

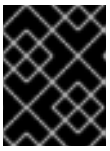
Java Security Manager で JBoss EAP を実行するには、起動時に **secmgr** オプションを使用する必要があります。これには 2 つの方法があります。

- 起動スクリプトでフラグを使用します。起動スクリプトで **-secmgr** フラグを使用するには、JBoss EAP インスタンスの起動時に、このフラグを含めます。

  起動スクリプトの例

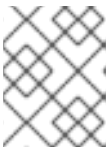
  ```
  ./standalone.sh -secmgr
  ```

- 起動設定ファイルの使用

  重要

  設定ファイルを編集する前に、ドメインまたはスタンドアロンサーバーを完全に停止する必要があります。

  注記

  マネージドドメインで JBoss EAP を使用している場合は、ドメインの各物理ホストまたはインスタンスで以下の手順を実行する必要があります。

  起動設定ファイルを使用して Java Security Manager を有効にするには、スタンドアロンインスタンスまたはマネージドドメインのいずれかを実行しているかに応じて、**standalone.conf** または **domain.conf** ファイルのいずれかを編集する必要があります。Windows で実行している場合は、代わりに **standalone.conf.bat** または **domain.conf.bat** ファイルが使用されます。

  設定ファイルの **secmgr="true"** 行のコメントを解除します。

  standalone.conf or domain.conf

  ```
  # Uncomment this to run with a security manager enabled
  SECMGR="true"
  ```

  standalone.conf.bat or domain.conf.bat

```
rem # Uncomment this to run with a security manager enabled
set "SECMGR=true"
```

5.4. 以前のバージョンからの移行に関する考慮事項

Java Security Manager を有効にして実行してい状態で、JBoss EAP の以前のバージョンから JBoss EAP 7 へアプリケーションを移動する場合は、ポリシーの定義方法と JBoss EAP 設定およびデプロイメントの両方に必要な設定における変更に注意してください。

5.4.1. ポリシーの定義

以前のリリースの JBoss EAP では、ポリシーは外部設定ファイルで定義されていました。JBoss EAP 7 では、ポリシーは **security-manager** サブシステムを使用し、デプロイメントに含まれる **Permissions.xml** または **jboss-permissions.xml** を使用して定義されます。両方を使用してポリシーを定義する方法の詳細は、前のセクション で説明しています。

5.4.2. JBoss EAP 設定の変更

これまでのバージョンの JBoss EAP では、JBoss EAP の起動中に **-Djava.security.manager** および **-Djava.security.policy** Java システムプロパティーを使用できました。これらはサポート外となりました。代わりに **secmgr** フラグを使用して JBoss EAP が Java Security Manager で実行できるようにする必要があります。**secmgr** フラグの詳細は、前のセクション で説明されています。

5.4.3. カスタムセキュリティーマネージャー

カスタムセキュリティーマネージャーは JBoss EAP 7 ではサポートされていません。

改訂日時: 2024-04-03