Red Hat JBoss Enterprise Application Platform 6.4

Identity Management の設定方法

Identity Management の設定方法

Last Updated: 2023-02-21

Red Hat JBoss Enterprise Application Platform 6.4 Identity Management の設定方法

Identity Management の設定方法

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

法律上の通知

概要

本ガイドの目的は、Red Hat JBoss Enterprise Application Platform 6 でユーザー ID と Identity Management に LDAP ディレクトリーと、その他の ID ストアを使用する方法のトピックを調査することです。本ガイドは、LDAP の基本と Red Hat JBoss Enterprise Application Platform 6 セキュリティーアーキテクチャーガイドでカバーされているその他の概念を拡張したものです。具体的には、このドキュメントは、管理インターフェイスでユーザー ID に LDAP を使用するように JBoss EAP 6 を設定するための実用的なガイドを提供し、Web アプリケーションで使用するためのセキュリティードメインを備えた LDAP およびその他の ID ストアを設定します。このドキュメントの対象は、管理インターフェイスとセキュリティードメインで使用するための LDAP サーバーおよびそ

の他の ID ストアの設定の入門書を探している JBoss EAP 6 管理者です。本ガイドを読む前に、ユーザーは Red Hat JBoss Enterprise Application Platform 6 セキュリティーアーキテクチャーガイドを読み、JBoss EAP 6 内のセキュリティー概念と、LDAP に関する基本的な知識をしっかりと理解している必要があります。また、本書では JBoss EAP 6 CLI インターフェースを使用して設定の変更を行います。スタンドアロンの JBoss EAP 6 インスタンスと JBoss EAP 6 ドメインの両方で CLI を使用する方法の詳細は、Red Hat JBoss Enterprise Application Platform 6 管理および設定ガイドの管理 CLI セクションを参照してください。本ドキュメントを完了すると、LDAP およびその他の ID ストアが JBoss EAP 6 でユーザー ID にどのように関連するかについての確かな基礎知識と、JBoss EAP 6 で アイデンティティー管理を設定する方法についての実践的な知識が得られます。

目次

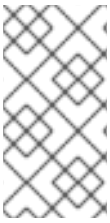
第1章 アイデンティティー管理の概要

アイデンティティー管理と ID ストアの背後にある基本は、Red Hat JBoss Enterprise Application Platform 6 Security Architecture guide ドキュメントの **Single Sign On (SSO)** セクションで説明されています。これらの概念は、SSO 外の JBoss EAP 6 管理インターフェイスおよび Web アプリケーションにセキュリティーを提供するためにも適用できます。具体的には、本ドキュメントでは、管理インターフェイス (LDAP) とセキュリティードメイン (LDAP、データベース、ファイルシステム) をセキュリティ保護するために、JBoss EAP 6 でアイデンティティー管理用のさまざまな ID ストアの設定について説明します。

第2章 LDAP の概要

LDAP の基本と基本的なユースケースは、Red Hat JBoss Enterprise Application Platform 6 Security Architecture guide ドキュメントの LDAP と Example Scenarios のセクションで説明されています。また、Security Subsystem セクションで、セキュリティードメインの LDAP 関連のログインモジュールについても説明します。

一般に、JBoss EAP 6 のコンテキストでは、LDAP はセキュリティードメインおよびセキュリティーレルムの ID ストアとして使用されます。これは、認証の決定を行うために使用できるだけでなく、JBoss EAP およびそれにデプロイされたアプリケーションの認証機関としても機能します。プリンシパルに関するさまざまなロールおよびアクセス情報は LDAP ディレクトリーに格納でき、JBoss EAP 6 で直接使用したり、既存の JBoss EAP 6 ロールにマップしたりできます。

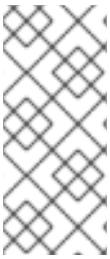


注記

データベースまたは他の外部データストアを使用して ID 情報を格納するのと同様に、データアクセスおよびデータストアと JBoss EAP 6 インスタンス間のデータ転送のパフォーマンスは、プリンシパルの認証と承認にパフォーマンスの影響を与える可能性があります。

第3章 LDAP での管理インターフェースのセキュア化

管理インターフェイスは、デフォルトで設定されているプロパティーファイルベースのセキュリティーレルムの代わりに、LDAP サーバー (Microsoft Active Directory を含む) に対して認証できます。これは、LDAP authenticator を使用して実行できます。LDAP オーセンティケーターは、最初にリモートディレクトリーサーバーへの接続 (アウトバウンド LDAP 接続を使用) を確立します。その後、ユーザーが認証システムに渡すユーザー名を使用して検索を実行し、LDAP レコードの完全修飾識別名 (DN) を探します。成功すると、クレデンシャルおよびユーザーによって提供されるパスワードとしてユーザーの DN で、LDAP サーバーへの新しい接続が確立されます。この 2 つ目の接続と LDAP サーバーへの認証に成功すると、DN が有効であることが検証され、認証に成功しました。



注記

LDAP による管理インターフェースのセキュリティー保護により、認証がダイジェストから BASIC/Plain に変更されます。これによりデフォルトで、ネットワーク上にユーザー名とパスワードが暗号化されずに送信されます。このトラフィックを暗号化し、この情報をクリアで送信しないようにするために、送信接続で SSL/TLS を有効に することができます。

3.1. 基本設定

LDAP Directory Server を管理インターフェイスの一部またはすべての認証ソースとして使用するには、次の手順を実行する必要があります。

1. LDAP サーバーへアウトバウンド接続を作成
2. 新しい LDAP 対応のセキュリティーレルムを作成し、アウトバウンド接続を参照
3. 管理インターフェースの新しいセキュリティーレルムを参照

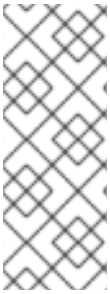
3.1.1. 1.LDAP サーバーへのアウトバウンド接続の作成

アウトバウンド LDAP 接続を作成する目的は、セキュリティーレルム (および JBoss EAP インスタンス) が LDAP サーバーへの接続を確立できるようにすることです。これは、セキュリティードメインの **Database** ログインモジュールで使用するデータソースを作成する場合と同様です。

LDAP アウトバウンド接続は以下の属性を許可します。

属性	必須	説明
url	○	ディレクトリーサーバーの URL アドレス。
search-dn	×	検索の実行が許可されているユーザーの完全識別名 (DN)。
search-credential	×	検索を実行する権限のあるユーザーのパスワード。

属性	必須	説明
initial-context-factory	×	接続を確立するときに使用する初期コンテキストファクトリー。デフォルトは **com.sun.jndi.ldap.LdapCtxFactory** です。
security-realm	×	接続の確立時に使用する設定済みの **SSLContext** を取得するために参照するセキュリティーレルム。
参考資料	×	検索時に参照が発生する場合の動作を指定します。有効なオプションは **IGNORE**、**FOLLOW**、および **THROW** です。**INGORE** (デフォルトの動作) は単に紹介を無視します。**FOLLOW**: 検索時に参照が見つかると、DirContext はその参照のフォローを試みます。ここでは、別のサーバーに接続するために同じ接続設定が使用できることを前提とし、参照機能で使用されている名前に到達できることを前提としています。**THROW** に設定すると、DirContext は例外 (**LdapReferralException**) を出力して、参照が必要であることを示します。これは、セキュリティーレルムが処理し、参照に使用する代替接続を識別しようとします。
handles-referrals-for	×	接続を処理できる参照情報を指定します。URI の一覧を指定する場合、それらはスペースで区切ります。これにより、複数の異なる認証情報をたどる必要がある場合に、接続プロパティーを持つ接続が定義され、使用されます。これは、別の認証情報が 2 つ目サーバーに対して認証する必要がある場合や、JBoss EAP 6 インストールから到達できない参照表現にサーバーが名前を返し、代替のアドレスを送信できる場合に役に立ちます。



注記

search-dn および **search-credential** は、ユーザーが提供するユーザー名とパスワードとは**異なります**。ここで提供される情報は、JBoss EAP インスタンスと LDAP サーバー間の初期接続を確立するために特に提供されます。この接続により、JBoss EAP は認証を試みるユーザーの DN を後続の検索することができます。(検索の結果となる) ユーザーの DN は認証を試み、入力したパスワードは認証プロセスを完了する際の第 2の (別の) 接続を確立するために使用されます。

LDAP サーバーの例を前提として、以下は、CLI コマンドとサーバーへのアウトバウンド LDAP 接続を設定するための結果の XML 設定です。

表3.1 LDAP サーバーの例

属性	値
url	127.0.0.1:389
search-credential	myPass
search-dn	cn=search,dc=acme,dc=com

アウトバウンド接続追加の CLI

```
/core-service=management/ldap-connection=ldap-connection/:add( \
search-credential=myPass, \
url=ldap://127.0.0.1:389, \
search-dn="cn=search,dc=acme,dc=com")
```

```
reload
```

結果の XML

```
<outbound-connections>
  <ldap name="ldap-connection"
        url="ldap://127.0.0.1:389"
        search-dn="cn=search,dc=acme,dc=com"
        search-credential="myPass"/>
</outbound-connections>
```

注記

上記の CLI コマンドは、JBoss EAP 6 のスタンドアロンインスタンスを想定して実行されました。JBoss EAP 6 ドメインでの CLI の使用の詳細は、Red Hat JBoss Enterprise Application Platform 6 Administration and Configuration Guide の **The Management CLI** セクションを参照してください。

注記

これにより、JBossEAP インスタンスと LDAP サーバーの間に **明確な** 接続が作成されます。SSL/TLS を使用して暗号化された接続を設定する方法の詳細については、「アウトバウンド LDAP 接続に SSL/TLS を使用」。

3.1.2. 2.新しい LDAP 対応セキュリティーレルムの作成

アウトバウンド LDAP 接続を作成したら、使用する新しい LDAP 対応セキュリティーレルムを作成する必要があります。

LDAP セキュリティーレルムには、以下の設定属性があります。

属性	説明
connection	LDAP ディレクトリーへの接続に使用する outbound-connections で定義された接続の名前。
base-dn	ユーザー検索を開始するためのコンテキストの識別名
再帰	検索が LDAP ディレクトリーツリー全体で再帰的であるか、指定したコンテキストのみを検索するか。デフォルトは false です。
user-dn	識別名を保持するユーザーの属性。この後、ユーザー完了時の認証テストに使用されます。デフォルトは dn です。
allow-empty-passwords	この属性は、空のパスワードを許可するかどうかを決定します。この属性のデフォルト値は false です。
username-attribute	username-attribute または advanced-filter のいずれかを指定する必要があります。ユーザーを検索する属性の名前。このフィルターは、ユーザーが入力したユーザー名が指定した属性と一致する単純な検索を実行します。
advanced-filter	username-attribute または advanced-filter のいずれかを指定する必要があります。提供されたユーザー ID に基づいてユーザーを検索するために使用される、完全に定義されたフィルター。この属性には、標準の LDAP 構文のフィルタークエリーが含まれます。フィルターには、{0} 形式の変数が含まれている必要があります。これは、ユーザーが指定したユーザー名で後ほど置き換えられます。**advanced-filter** の使用に関する詳細と例 は、LDAP と RBAC の認証の組み合わせセクションの <advanced-filter> の部分にあります。

警告

空の LDAP パスワードは、セキュリティ上の懸念が大きいため許可されていません この動作が環境内で特に必要でない限り、空のパスワードは許可されず、**allow-empty-passwords** は false のままになるようにしてください。

ldap-connection アウトバウンド LDAP 接続を使用して LDAP 対応のセキュリティーレルムを設定するための CLI コマンドと結果の XML 設定を次に示します。

CLI

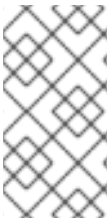
```
/core-service=management/security-realm=ldap-security-realm:add
```

```
/core-service=management/security-realm=ldap-security-realm/authentication=ldap:add( \
connection="ldap-connection", base-dn="cn=users,dc=acme,dc=com", \
username-attribute="sambaAccountName")
```

```
reload
```

結果の XML

```
<security-realm name="ldap-security-realm">
  <authentication>
    <ldap connection="ldap-connection" base-dn="cn=users,dc=acme,dc=com">
      <username-filter attribute="sambaAccountName"/>
    </ldap>
  </authentication>
</security-realm>
```



注記

上記の CLI コマンドは、JBoss EAP 6 のスタンドアロンインスタンスを想定して実行されました。JBoss EAP 6 ドメインでの CLI の使用の詳細は、Red Hat JBoss Enterprise Application Platform 6 Administration and Configuration Guide の **The Management CLI** セクションを参照してください。

3.1.3. 3.管理インターフェースの新しいセキュリティーレルムを参照

セキュリティーレルムが作成され、アウトバウンド LDAP 接続を使用している場合、新しいセキュリティーレルムは管理インターフェースで参照される必要があります。

HTTP インターフェイスを更新するための CLI コマンド

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

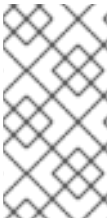
ネイティブインターフェイスを更新するための CLI コマンド

-

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

結果の XML

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



注記

上記の CLI コマンドは、JBoss EAP 6 のスタンドアロンインスタンスを想定して実行されました。JBoss EAP 6 ドメインでの CLI の使用の詳細は、Red Hat JBoss Enterprise Application Platform 6 Administration and Configuration Guide の **The Management CLI** セクションを参照してください。

3.2. アウトバウンド LDAP 接続に SSL/TLS を使用

JBoss EAP 6 は、SSL/TLS を使用した LDAP サーバーへのアウトバウンド接続を使用するように設定できます。SSL/TLS で保護されたアウトバウンド LDAP 接続を作成するには、次の手順を実行する必要があります。

1. アウトバウンド LDAP 接続が使用するキーストアとトラストストアを使用してセキュリティーレルムを設定
2. SSL/TLS URL およびセキュリティーレルムを使用してアウトバウンド LDAP 接続を作成 .
3. 管理インターフェースで使用するためにアウトバウンド ldap 接続を使用する新しいセキュリティーレルムを作成.

3.2.1. 1.使用するアウトバウンド LDAP 接続のセキュリティーレルムを設定

セキュリティーレルムには、JBoss EAP 6 サーバーがそれ自体と LDAP サーバー間の通信の復号化/暗号化に使用するキーで設定されたキーストアが含まれる必要があります。このキーストアにより、JBoss EAP 6 インスタンスは LDAP サーバーに対して自己検証することもできます。セキュリティーレルムには、LDAP サーバーの証明書が含まれるトラストストア、(または LDAP サーバーの証明書の署名に使用する認証局の証明書が含まれる必要もあります)。キーストアとトラストストアの設定や、これらを使用するセキュリティーレリムの作成の説明は、JBoss EAP 6 サーバーセキュリティーの設定方法ガイドの管理インターフェースの双方向 SSL/TLS の設定を参照してください。

3.2.2. 2.SSL/TLS URL およびセキュリティーレルムを使用してアウトバウンド LDAP 接続を作成

LDAP サーバーへのアウトバウンド接続の作成 で定義されたプロセスと同様に、アウトバウンド LDAP 接続を作成する必要がありますが、LDAP サーバーの SSL/TLS URL と SSL/TLS セキュリティーレルムを使用します。

LDAP サーバーのアウトバウンド LDAP 接続および SSL/TLS セキュリティーレルムを作成したら、その情報でアウトバウンド LDAP 接続を更新する必要があります。

SSL/TLS URL でアウトバウンド接続を追加する CLI の例

```
/core-service=management/ldap-connection=ldap-connection/:add( \
search-credential=myPass, \
url=ldaps://LDAP_HOST:LDAP_PORT, \
search-dn="cn=search,dc=acme,dc=com")
```

SSL/TLS 証明書を使用したセキュリティーレルムの追加

```
/core-service=management/ldap-connection=ldap-connection:write-attribute( \
name=security-realm,value="CertificateRealm")
```

```
reload
```

結果の XML

```
<outbound-connections>
  <ldap name="ldap-connection" url="ldaps://LDAP_HOST:LDAP_PORT"
  security-realm="CertificateRealm"
  search-dn="cn=search,dc=acme,dc=com"
  search-credential="myPass" />
</outbound-connections>
```

注記

上記の CLI コマンドは、JBoss EAP 6 のスタンドアロンインスタンスを想定して実行されました。JBoss EAP 6 ドメインでの CLI の使用の詳細は、Red Hat JBoss Enterprise Application Platform 6 Administration and Configuration Guide の **The Management CLI** セクションを参照してください。

3.2.3. 3.管理インターフェースで使用するためにアウトバウンド LDAP 接続を使用する新しいセキュリティーレルムを作成

Creating a new LDAP-Enabled Security Realm セクションで概説されている手順に従い、管理インターフェイスで新しいセキュリティーレルムを参照します。

3.3. LDAP および RBAC

RBAC (Role-Based Access Control) は、一連の管理ユーザーに一連の権限 (ロール) を指定するメカニズムであり、ユーザーに完全な無制限のアクセスを許可することなく、さまざまな管理責任を付与できます。RBAC の詳細は、Red Hat JBoss Enterprise Application Platform 6 Security Architecture guide の **Role-Based Access Control** セクションを参照してください。

RBAC は承認に使用され、認証は個別に処理されます。LDAP は認証だけでなく承認にも使用できるため、JBoss EAP 6 は、LDAP なしの認証に RBAC を使用するように設定できます (LDAP またはその他の認証メカニズムを使用)。JBoss EAP 6 は、管理インターフェイスで承認を決定するために LDAP と組み合わせた RBAC を使用するように設定することもできます。

3.3.1. LDAP および RBAC の使用による非依存

JBoss EAP 6 では、認証および承認をセキュリティーレルムで個別に設定できます。これにより、LDAP を認証方法として設定でき、RBAC を承認メカニズムとして設定できます。この方法で設定された場合、ユーザーが管理インターフェースへのアクセス試行すると、最初に設定された LDAP サーバーを使用して認証されます。成功すると、(LDAP サーバーにあるグループ情報に関係なく、)ユーザーロール (および、そのロールが設定されたパーミッション) は RBAC のみを使用して決定されます。

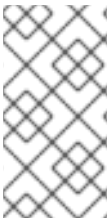
管理インターフェイスの承認メカニズムとして RBAC のみを使用する方法の詳細は、Server Security guide を参照してください。管理インターフェイスを使用した認証用に LDAP を設定する方法の詳細は、このトピックに関する 前のセクションを参照してください。

3.3.2. LDAP と RBAC の承認の統合

LDAP サーバー (またはプロパティーファイル) を使っ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、単一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 は、このグループ情報を使用してロールをユーザーに自動的に割り当てるか、ロールからユーザーを除外するように設定できます。

LDAP ディレクトリーには、ユーザーアカウントおよびグループのエントリーが含まれ、属性によって参照されます。LDAP サーバー設定によっては、ユーザーエンティティーはユーザーが所属するグループを **memberOf** 属性を介してマップできます。また、グループエンティティーは、**uniqueMember** 属性によってユーザーが属するユーザーをマップできます。ユーザーが LDAP サーバーに正常に認証されると、グループ検索が実行され、そのユーザーのグループ情報が読み込まれます。使用している Directory Server に応じて、グループ検索は、単純な名前 (通常は認証に使用されるユーザー名) を使用するか、ディレクトリー内のユーザーのエントリーの識別名を使用して実行できます。LDAP をセキュリティーレルムの承認メカニズムとして設定すると、グループ検索 (**<group-search>**) およびユーザー名と識別名 (**<username-to-dn>**) 間のマッピングが設定されます。

ユーザーのグループメンバーシップ情報が LDAP サーバーから決定されると、RBAC 設定内のマッピングを使用して、ユーザーが所有するロールが決定されます。このマッピングは、グループおよび個々のユーザーを明示的に包含または除外するように設定されます。

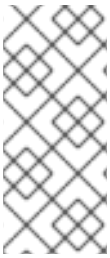


注記

サーバーに接続するユーザーの認証手順が常に最初に行われます。ユーザーの認証に成功すると、サーバーはユーザーのグループを読み込みます。認証のステップと承認の手順では、それぞれ LDAP サーバーへの接続が必要です。セキュリティーレルムは、グループの読み込みステップで認証接続を再利用してこのプロセスを最適化します。

3.3.2.1. group-search の使用

グループメンバーシップ情報を検索する場合に使用できるスタイルには、**Principal to Group** および **Group to Principal** があります。グループのプリンシパルには、メンバーであるグループ (つまり、**memberOf** 属性) への参照を含むユーザーのエントリーがあります。Group to Principal には、グループのメンバーであるユーザー (つまり、**uniqueMember**) への参照が含まれるグループのエントリーがあります。



注記

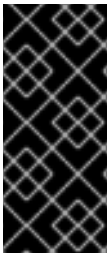
JBoss EAP 6 は Principal と Group to Principal の両方をサポートしますが、Group to Principal で Principal を使用することが推奨されます。グループへのプリンシパルが使用される場合、検索を実行せずに既知の識別名の属性を読み取ることで、グループ情報を直接読み込むことができます。Group to Principal では、ユーザーを参照するすべてのグループを特定するために、幅広い検索が必要になります。

Principal to Group と Group to Principal はいずれも、以下の属性を含む **<group-search>** タグを使用します。

属性	説明
group-name	この属性は、ユーザーがメンバーとなるグループの一覧として返されたグループ名に使用されるフォームを指定するために使用されます。これは、グループ名の単純な形式か、グループの識別名のいずれかになります。識別名が必要な場合は、この属性を DISTINGUISHED_NAME に設定できます。デフォルトは SIMPLE です。
iterative	この属性は、ユーザーがメンバーとなっているグループを特定した後、そのグループがメンバーとなっているグループを特定するために、グループに基づいて繰り返し検索を行う必要があるかどうかを示します。反復検索が有効になっていると、他のグループまたはサイクルが検出された場合に、メンバーではないグループに到達するまで継続されます。デフォルトは false です。
group-dn-attribute	属性が識別名であるグループのエントリーデフォルトは dn です。
group-name-attribute	属性が単純な名前であるグループのエントリーデフォルトは uid です。

注記

同時グループメンバーシップは問題ではありません。各検索の記録は、すでに検索されているグループが再度検索されないように保持されます。



重要

反復検索が機能するには、グループエントリーがユーザーエントリーと同じである必要があります。ユーザーがメンバーとなっているグループを識別するのに使用するのと同じアプローチを使用して、グループがメンバーとなっているグループを特定します。これは、グループメンバーシップ、クロス参照に使用される属性名の変更、または参照方向が変更された場合は利用できません。

グループ検索の Principal to Group (memberOf)

たとえば、**GroupOne** のメンバーの **TestUserOne** ユーザーと、**GroupOne** が次に **GroupFive** のメンバーとなる場合を考えます。グループメンバーシップは、メンバーレベルで **memberOf** 属性を使用することで表示されます。これは、**TestUserOne** によって、**memberOf** 属性が **GroupOne** の **dn** に設定されることになります。**GroupOne** は次に **memberOf** 属性を **GroupFive** の **dn** に設定します。

このタイプの検索を使用するために、**principal-to-group** 要素が **group-search** 要素に追加されます。

Principal to Group (つまり memberOf) の設定

```
<security-realm name="ldap-security-realm">
  <authentication>
    <ldap connection="ldap-connection" >
      <!-- configuration -->
    </ldap>
  </authentication>
  <authorization>
    <ldap connection="ldap-connection">
      <!-- configuration -->
        <group-search group-name="SIMPLE" iterative="true" group-dn-attribute="dn" group-name-
attribute="uid">
            <principal-to-group group-attribute="memberOf" />
        </group-search>
    </ldap>
  </authorization>
</security-realm>
```

<group-search> 内に、**<principal-to-group>** が **group-attribute** 属性で追加されていることに注意してください。参考情報:

表3.2 principal-to-group

属性	説明
group-attribute	ユーザーがメンバーとなっているグループの識別名と一致するユーザーエントリーの属性の名前。デフォルトは memberOf です。
prefer-original-connection	この値は、参照に従う際に優先するグループ情報を示します。プリンシパルがロードされるたびに、各グループメンバーシップの属性がその後にロードされます。属性がロードされるたびに、最後の参照情報からの元の接続または接続のいずれかを使用できます。デフォルトは **true** です。

Group to Principal (つまり uniqueMember) のグループ検索

Principal to Group と同じ例を考えてみましょう。ここでは、**GroupOne** のメンバーであるユーザー**TestUserOne**、次に **GroupOne** が **GroupFive** のメンバーです。ただし、この場合、グループメンバーシップはグループレベルで設定された **uniqueMember** 属性を使用することで表示されます。これは **GroupFive** によって、**uniqueMember** が **GroupOne** の **dn** に設定されていることを意味します。**GroupOne** は次に、**uniqueMember** を **TestUserOne** の **dn** に設定します。

このタイプの検索を使用するために、**group-to-principal** 要素が **group-search** 要素に追加されます。

Group to Principal (つまり uniqueMember) の設定

```
<security-realm name="ldap-security-realm">
  <authentication>
    <ldap connection="ldap-connection" >
      <!-- configuration -->
    </ldap>
  </authentication>
  <authorization>
```

```
    <ldap connection="ldap-connection">
      <!-- configuration -->
      <group-search group-name="SIMPLE" iterative="true" group-dn-attribute="dn" group-name-
attribute="uid">
        <group-to-principal base-dn="ou=groups,dc=group-to-principal,dc=example,dc=org"
recursive="true" search-by="DISTINGUISHED_NAME">
          <membership-filter principal-attribute="uniqueMember" />
        </group-to-principal>
      </group-search>
    </ldap>
  </authorization>
</security-realm>
```

<group-search> 内で、**<group-to-principal>** に属性が追加され、**<membership-filter>** が含まれていることに注意してください。**group-to-principal** は、ユーザーエントリーを参照するグループの検索が実行される方法を定義するために使用され、**membership-filter** は、相互参照を定義するために使用されます。

参考情報:

表3.3 group-to-principal

属性	説明
base-dn	検索を開始するために使用するコンテキストの識別名。
recursive	サブコンテキストも検索されるかどうか。デフォルトは false です。
search-by	検索で使用するロール名の形式。有効な値は SIMPLE および DISTINGUISHED_NAME です。デフォルトは DISTINGUISHED_NAME です。
prefer-original-connection	この値は、参照に従う際に優先するグループ情報を示します。プリンシパルがロードされるたびに、各グループメンバーシップの属性がその後にロードされます。属性がロードされるたびに、最後の参照情報からの元の接続または接続のいずれかを使用できます。

表3.4 membership-filter

属性	説明
principal-attribute	ユーザーエントリーを参照するグループエントリーの属性の名前。デフォルトは member です。

3.3.2.2. username-to-dn の使用

承認セクション内でルールを定義して、ユーザーの簡単なユーザー名を識別名に変換できます。username-to-dn 要素は、ユーザー名を LDAP ディレクトリー内のエントリーの識別名にマップする方法を指定します。この要素は**任意**で、以下のいずれかが true の場合のみ必要になります。

- 認証および承認の手順は、異なる LDAP サーバーに対して行われます。
- グループ検索は、識別名を使用します。

注記

これは、セキュリティーレルムが LDAP および Kerberos 認証の両方をサポートし、認証中に検出された DN を実行している場合は、Kerberos に変換が必要な場合でも適用可能です。

これには、以下の属性が含まれます。

表3.5 username-to-dn

属性	説明
force	force 属性が「false」に設定されている場合には、認証中の **ユーザー名から識別名のマッピング**の検索結果はキャッシュされ、承認クエリー時に再利用されます。force が true の場合は、(グループの読み込み時) 承認中に検索が再度実行されます。これは通常、異なるサーバーが認証および承認を実行すると実行されます。

username-to-dn は以下のいずれかで設定できます。

<username-is-dn>

リモートユーザーが入力するユーザー名がユーザーの識別名であることを指定します。

username-is-dn の例

```
<security-realm name="ldap-security-realm">
  <authentication>
    <ldap connection="ldap-connection" >
      <!-- configuration -->
    </ldap>
  </authentication>
  <authorization>
    <ldap connection="ldap-connection">
      <username-to-dn force="false">
        <username-is-dn />
      </username-to-dn>
      <!-- configuration -->
    </ldap>
  </authorization>
</security-realm>
```

これにより 1:1 マッピングが定義され、追加設定はありません。

<username-filter>

指定された属性で、指定のユーザー名に対して一致するものが検索されます。

username-filter の例

```
<security-realm name="ldap-security-realm">
  <authentication>
    <ldap connection="ldap-connection" >
      <!-- configuration -->
    </ldap>
  </authentication>
  <authorization>
    <ldap connection="ldap-connection">
      <username-to-dn force="true">
        <username-filter base-dn="dc=people,dc=harold,dc=example,dc=com" recursive="false"
attribute="sn" user-dn-attribute="dn" />
      </username-to-dn>
      <!-- configuration -->
    </ldap>
  </authorization>
</security-realm>
```

属性	説明
base-dn	検索を開始するコンテキストの識別名。
recursive	検索がサブコンテキストに広がるかどうか。デフォルトは false です。
属性	提供されたユーザー名と照合するユーザーのエントリーの属性。デフォルトは uid です。
user-dn-attribute	ユーザーの識別名を取得するために読み取る属性。デフォルトは dn です。

<advanced-filter>

このオプションは、カスタムフィルターを使用して、ユーザーの識別名を特定します。

advanced-filter の例

```
<security-realm name="ldap-security-realm">
  <authentication>
    <ldap connection="ldap-connection" >
      <!-- configuration -->
    </ldap>
  </authentication>
  <authorization>
    <ldap connection="ldap-connection">
      <username-to-dn force="true">
        <advanced-filter base-dn="dc=people,dc=harold,dc=example,dc=com" recursive="false"
filter="sAMAccountName={0}" user-dn-attribute="dn" />
      </username-to-dn>
      <!-- configuration -->
```

```
    </ldap>
  </authorization>
</security-realm>
```

username-filter の例にある属性に一致する属性の場合、意味とデフォルト値は同じです。追加の属性があります。

属性	説明
filter	ユーザー名が {0} プレースホルダーで置換されるユーザーのエントリーの検索に使用されるカスタムフィルター。

重要

XML はフィルターの定義後も有効でなければならないので、特殊文字 (&など) が使用されている場合は、適切な形式であることを確認してください。たとえば **&** 文字の場合は **&** です。

3.3.2.3. LDAP グループ情報の RBAC ロールへのマッピング

LDAP サーバーへの接続が作成され、グループ検索が適切に設定されたら、LDAP グループと RBAC ロール間のマッピングを作成する必要があります。このマッピングは包括的かつ排他的で可能で、ユーザーはグループメンバーシップ情報に基づいて自動的にロールを割り当てることができます。

警告

RBAC が設定されていない場合は、注意が必要です。、特に新規に作成された LDAP 対応レルムに切り替える場合は十分に注意してください。ユーザーとロールを適切に設定せずに RBAC を有効にすると、管理者が管理インターフェースにログインできなくなることがあります。

注記

以下の CLI コマンドは、JBoss EAP 6 のスタンドアロンインスタンスを想定して実行されました。JBoss EAP 6 ドメインでの CLI の使用の詳細は、Red Hat JBoss Enterprise Application Platform 6 Administration and Configuration Guide の **The Management CLI** セクションを参照してください。

RBAC が有効になっており、設定されていることの確認

LDAP と RBAC ロール間のマッピングを使用するには、RBAC を有効化し、初期設定する必要があります。

```
/core-service=management/access=authorization:read-attribute(name=provider)
```

以下の結果が出されます。

```
{
    "outcome" => "success",
    "result" => "rbac"
}
```

RBAC の有効化と設定の詳細は、Red Hat JBoss Enterprise Application Platform 6 How to Configure Server Security guide の **Enabling Role-Based Access Control** セクションを参照してください。

既存のロール一覧の確認

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)
```

ロールの一覧を作成するもの

```
{
    "outcome" => "success",
    "result" => [
        "Administrator",
        "Deployer",
        "Maintainer",
        "Monitor",
        "Operator",
        "SuperUser"
    ]
}
```

また、ロールの既存のすべてのマッピングを確認できます。

```
/core-service=management/access=authorization/role-mapping=Administrator: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"
            }
```

```
        }
    }
}
```

Role-Mapping エントリーの設定

ロールに Role-Mapping エントリーがない場合は、これを作成する必要があります。たとえば、以下のようになります。

```
/core-service=management/access=authorization/role-mapping=Auditor:read-resource()
```

```
{
    "outcome" => "failed",
    "failure-description" => "JBAS014807: Management resource '[
    (\"core-service\" => \"management\"),
    (\"access\" => \"authorization\"),
    (\"role-mapping\" => \"Auditor\")
]' not found"
}
```

ロールマッピングを追加する:

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

```
{
    "outcome" => "success"
}
```

検証する:

```
/core-service=management/access=authorization/role-mapping=Auditor:read-resource()
```

```
{
    "outcome" => "success",
    "result" => {
        "include-all" => false,
        "exclude" => undefined,
        "include" => undefined
    }
}
```

除外および除外のロールへのグループの追加

ロールに含める、またはロールから除外するためにグループを追加できます。

注記

除外マッピングが優先されるか、または包含マッピングが優先されます。

包含にグループを追加:

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

除外するグループを追加:

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

結果の確認:

```
[standalone@localhost:9999 /] /core-service=management/access=authorization/role-mapping=Auditor:read-resource(recursive=true)
```

```
{
    "outcome" => "success",
    "result" => {
        "include-all" => false,
        "exclude" => {"group-GroupToExclude" => {
            "name" => "GroupToExclude",
            "realm" => undefined,
            "type" => "GROUP"
        }},
        "include" => {"group-GroupToInclude" => {
            "name" => "GroupToInclude",
            "realm" => undefined,
            "type" => "GROUP"
        }}
    }
}
```

ロールからのグループの削除

除外または含めるために追加されたグループも削除される場合があります

グループを追加から削除するには、以下を使用します。

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

除外からグループを削除するには、以下を使用します。

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

3.4. キャッシュの有効化

セキュリティーレルムは、認証とグループローディングの両方で LDAP クエリーの結果をキャッシュする機能も提供します。これにより、特定の状況で複数の検索で異なるクエリーの結果を再利用できます (たとえば、グループのグループメンバーシップ情報を繰り返しクエリーするなど)。利用できるキャッシュは 3 種類あり、それぞれは個別に設定され、独立して動作します。

- authentication

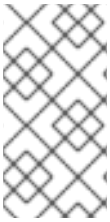
- group-to-principal
- username-to-dn

3.4.1. キャッシュの設定

キャッシュが相互に独立している場合でも、3 つすべてのキャッシュは同じ方法で設定されます。各キャッシュは以下の設定オプションを提供します。

属性	説明
type	これは、キャッシュが準拠するエビクションストラテジーを定義します。オプションは **by-access-time** および **by-search-time** です。**by-access-time** は、最後のアクセスから一定期間経過した後にキャッシュから項目をエビクトします。**by-search-time** は、最後のアクセスに関係なく、キャッシュにある期間に基づいて項目をエビクトします。
eviction-time	これはストラテジーに応じてエビクションに使用される時間 (秒単位) を定義します。
cache-failures	これは、失敗した検索のキャッシングを有効/ 無効にするブール値です。これにより、LDAP サーバーが、失敗した同じ検索によって繰り返しアクセスされてしまうのを防げる可能性がありますが、存在しないユーザーの検索でキャッシュがいっぱいに成ってしまう可能性もあります。この設定は、認証キャッシュで特に重要になります。
max-cache-size	これは、キャッシュの最大サイズ (項目数) を定義します。これは、アイテムがエビクトされるタイミングを指定します。古い項目は、新規認証のスペースを確保するためにキャッシュからエビクトされ、必要に応じて検索が行われます。つまり、**max-cache-size** は新規認証の試行や検索を阻止しません。

3.4.2. 例



注記

この例では、既存の LDAP サーバーに接続し、認証と承認用に設定された **LDAPRealm** という名前のセキュリティーレルムが作成されていることを前提としています。現在の設定を表示するコマンドについては、後のセクションで詳しく説明します。LDAP を使用するセキュリティーレルムの作成の詳細については、こちらを参照してください。

ベース設定の例

```
"core-service" : {
        "management" : {
            "security-realm" : {
```

```
            "LDAPRealm" : {
                "authentication" : {"ldap" : {
                    "allow-empty-passwords" : false,
                    "base-dn" : "...",
                    "connection" : "MyLdapConnection",
                    "recursive" : false,
                    "user-dn" : "dn",
                    "username-attribute" : "uid",
                    "cache" : null
                }},
                "authorization" : {"ldap" : {
                    "connection" : "MyLdapConnection",
                    "group-search" : {"group-to-principal" : {
                        "base-dn" : "...",
                        "group-dn-attribute" : "dn",
                        "group-name" : "SIMPLE",
                        "group-name-attribute" : "uid",
                        "iterative" : true,
                      "principal-attribute" : "uniqueMember",
                        "search-by" : "DISTINGUISHED_NAME",
                        "cache" : null
                    }},
                    "username-to-dn" : {"username-filter" : {
                        "attribute" : "uid",
                        "base-dn" : "...",
                        "force" : false,
                        "recursive" : false,
                        "user-dn-attribute" : "dn",
                        "cache" : null
                    }}
                }},
            }
        }
    }
```

"cache" : null が表示されるすべてのエリアで、キャッシュを設定できます。

認証

認証時に、ユーザーの識別名はこの定義を使用して検出されます。また、LDAP サーバーへの接続が試行され、その ID がこれらの認証情報を使用して作成されます。

group-search 定義

グループ検索の定義があります。この場合は反復検索です (上記のサンプル設定では **iterative** が **true** に設定されているため)。まず、ユーザーが直接メンバーとなっているすべてのグループを見つける検索が実行されます。その後、これらの各グループに検索が実行され、他のグループにメンバーシップがあるかどうかが特定されます。このプロセスは、cyclic 参照が検出されるまで、または最終グループが他のグループのメンバーではないまで継続されます。

グループ検索の username-to-dn 定義

グループ検索は、ユーザーの識別名の可用性に依存します。このセクションはすべての状況で使用されるわけではありませんが、ユーザーの識別名の検出試行に使用することができます。これは、たとえば、2 番目の形式の認証 (ローカル認証など) がサポートされている場合に役立つか、必要になる場合があります。

3.4.2.1. 現在のキャッシュ設定の読み取り

注記

本セクションおよび後続のセクションで使用される CLI コマンドは、セキュリティーレルムの名前に **LDAPRealm** を使用します。これは、設定する実際のレルムの名前に置き換えてください。

現在のキャッシュ設定を読み取る CLI コマンド

```
/core-service=management/security-realm=LDAPRealm:read-resource(recursive=true)
```

出力

```
{
    "outcome" => "success",
    "result" => {
        "map-groups-to-roles" => true,
        "authentication" => {"ldap" => {
            "advanced-filter" => undefined,
            "allow-empty-passwords" => false,
            "base-dn" => "dc=example,dc=com",
            "connection" => "ldapConnection",
            "recursive" => true,
            "user-dn" => "dn",
            "username-attribute" => "uid",
            "cache" => undefined
        }},
        "authorization" => {"ldap" => {
            "connection" => "ldapConnection",
            "group-search" => {"principal-to-group" => {
                "group-attribute" => "description",
                "group-dn-attribute" => "dn",
                "group-name" => "SIMPLE",
                "group-name-attribute" => "cn",
                "iterative" => false,
                "prefer-original-connection" => true,
                "skip-missing-groups" => false,
                "cache" => undefined
            }},
            "username-to-dn" => {"username-filter" => {
                "attribute" => "uid",
                "base-dn" => "ou=Users,dc=jboss,dc=org",
                "force" => true,
                "recursive" => false,
                "user-dn-attribute" => "dn",
                "cache" => undefined
            }}
        }},
        "plug-in" => undefined,
        "server-identity" => undefined
    }
}
```

3.4.2.2. キャッシュの有効化

注記

このセクションおよび後続のセクションで使用される CLI コマンドは、セキュリティーレルムの認証部分 (つまり、**/authentication=ldap/**) のキャッシュを設定します。承認部分のキャッシュは、コマンドのパスを更新することで、同様の方法で設定することもできます。

キャッシュを有効にするための CLI

```
/core-service=management/security-realm=LDAPRealm/authentication=ldap/cache=by-access-time:add( \
eviction-time=300, cache-failures=true, max-cache-size=100)
```

このコマンドは、エビクション時間が 300 秒 (5 分) で最大キャッシュサイズが 100 の認証に、**by-access-time** キャッシュを追加します。さらに、失敗した検索はキャッシュされます。または、**by-search-time** キャッシュを設定することもできます。

```
/core-service=management/security-realm=LDAPRealm/authentication=ldap/cache=by-search-time:add( \
eviction-time=300, cache-failures=true, max-cache-size=100)
```

3.4.2.3. 既存キャッシュの検査

既存のキャッシュを検査するための CLI

```
/core-service=management/security-realm=LDAPRealm/authentication=ldap/cache=by-access-time:read-resource(include-runtime=true)
```

出力

```
{
    "outcome" => "success",
    "result" => {
        "cache-failures" => true,
        "cache-size" => 1,
        "eviction-time" => 300,
        "max-cache-size" => 100
    }
}
```

include-runtime 属性は、キャッシュ内のアイテムの現在の数 (この場合は 1) を表示する **cache-size** 要素を追加します。

3.4.2.4. 既存のキャッシュの内容のテスト

既存のキャッシュの内容をテストするための CLI

```
/core-service=management/security-realm=LDAPRealm/authentication=ldap/cache=by-access-time:contains(name=TestUserOne)
```

出力

-

```
{
    "outcome" => "success",
    "result" => true
}
```

これは、**TestUserOne** のエントリーがキャッシュに存在することを示しています。

3.4.2.5. キャッシュのフラッシュ

キャッシュからの個々のアイテムと同様に、キャッシュ全体がフラッシュされる場合があります。

単一アイテムをフラッシュするための CLI

```
/core-service=management/security-realm=LDAPRealm/authentication=ldap/cache=by-access-time:flush-cache(name=TestUserOne)
```

キャッシュ全体をフラッシュするための CLI

```
/core-service=management/security-realm=LDAPRealm/authentication=ldap/cache=by-access-time:flush-cache()
```

3.4.2.6. キャッシュの削除

キャッシュを削除するための CLI

```
/core-service=management/security-realm=LDAPRealm/authentication=ldap/cache=by-access-time:remove()
```

```
reload
```

第4章 LDAP を使用するためのセキュリティードメインの設定

セキュリティードメインは、ログインモジュールを使用して認証および承認に LDAP サーバーを使用するよう設定できます。セキュリティードメインとログインモジュールの基本は、Red Hat JBoss Enterprise Application Platform 6 Security Architecture guide で説明されています。**LdapExtended** は LDAP サーバー (Active Directory を含む) との統合に推奨されるログインモジュールですが、他のいくつかの LDAP ログインモジュールも使用できます。特に、LDAP を使用するようセキュリティードメインを設定するには、**Ldap**、**AdvancedLdap**、**AdvancedAdLdap** を使用することもできます。本セクションでは、**LdapExtended** ログインモジュールを使用して、認証および承認に LDAP を使用するセキュリティードメインの作成方法を説明しますが、他の LDAP ログインモジュールを使用することもできます。その他の LDAP ログインモジュールの詳細は、Red Hat JBoss Enterprise Application Platform 6 Security Guide を参照してください。

4.1. LDAPEXTENDED ログインモジュール

LdapExtended (org.jboss.security.auth.spi.LdapExtLoginModule) はログインモジュール実装で、LDAP サーバー上のバインドユーザーと関連付けられたロールの特定に使用されます。ロールは再帰的にクエリーを行い、DN に従って階層的なロール構造を移動します。セキュリティードメインで LDAP を使用する場合、ほとんどの場合、特に Active Directory 以外の LDAP 実装では、**LdapExtedned** ログインモジュールを使用する必要があります。

表4.1 完全な LdapExtended ログインモジュール設定オプション

オプション	Type	デフォルト	説明
java.naming.factory.initial	クラス名	com.sun.jndi.ldap.LdapCtxFactory	InitialContextFactory 実装クラス名。
java.naming.provider.url	ldap:// URL	Java.naming.security.protocol の値が SSL、ldap://localhost:636 の場合、それ以外は ldap://localhost:389	LDAP サーバーの URL。
java.naming.security.authentication	SASL メカニズムの none、simple、または name	デフォルトは **simple** です。プロパティーが明示的に定義されていない場合、この動作はサービスプロバイダーによって決定されます。	LDAP サーバーにバインドするために使用するセキュリティーレベル。
java.naming.security.protocol	トランスポートプロトコル	指定されていない場合、プロバイダーによって決定されます。	SSL などのセキュアなアクセスに使用するトランスポートプロトコル。
baseCtxDN	完全修飾 DN	none	ユーザーの検索を開始するため、トップレベルのコンテキストの固定 DN です。
bindCredential	文字列、オプションで暗号化	none	DN の認証情報を保存するために使用されます。

オプション	Type	デフォルト	説明
bindDN	完全修飾 DN	none	ユーザーおよびロールクエリーの LDAP サーバーに対してバインドするために使用される DN です。この DN には、baseCtxDN および rolesCtxDN 値の読み取りおよび検索パーミッションが必要です。
baseFilter	LDAP フィルター文字列	none	認証するユーザーのコンテキストを見つけるために使用される検索フィルター。{0} 式を使用しているフィルターに、入力ユーザー名、またはログインモジュールコールバックから取得した userDN が置換されます。検索フィルターの一般的な例は (uid={0}) です。
rolesCtxDN	完全修飾 DN	none	ユーザーロールを検索するためのコンテキストの固定 DN です。これは、実際のロールがである DN ではなく、ユーザーロールを含むオブジェクトがある DN です。たとえば、Microsoft Active Directory サーバーでは、ユーザーアカウントが DN になります。

オプション	Type	デフォルト	説明
roleFilter	LDAP フィルター文字列	none	認証済みユーザーと関連付けられたロールを検索するために使用される検索フィルター。{0} 式を使用しているフィルターに、入力ユーザー名、またはログインモジュールコールバックから取得した userDN が置換されます。認証済み userDN は {1} が使用されたフィルターに置き換えられます。入力ユーザー名に一致する検索フィルター例は (member={0}) です。認証済み userDN に一致する他の例は (member={1}) です。
roleAttributeID	attribute	roles	ユーザーロールを含む属性の名前。
roleAttributeIsDN	true または false	false	roleAttributeID にロールオブジェクトの完全修飾 DN が含まれるかどうか。false の場合は、コンテキスト名の roleNameAttributeId 属性の値からこのロール名が取得されます。Microsoft Active Directory などの特定のディレクトリースキーマでは、この属性を true に設定する必要があります。
defaultRole	ロール名	none	すべての認証ユーザーに含まれるロール

オプション	Type	デフォルト	説明
parseRoleNameFromDN	true または false	false	クエリーによって返された DN に roleNameAttributeID が含まれるかどうかを示すフラグ。true に設定した場合には、DN は roleNameATtributeID に対してチェックされます。false に設定すると、DN は roleNameATtributeID に対して確認されません。このフラグは LDAP クエリーのパフォーマンスを向上できます。
parseUsername	true または false	false	DN がユーザー名に対して解析されるかどうかを示すフラグ。true に設定した場合には、 DN はユーザー名に対して解析されます。false に設定した場合には、 DN はユーザー名に対して解析されません。このオプションは、usernameBeginString および usernameEndString とともに使用されます。
usernameBeginString	string	none	DN の最初から削除される文字列を定義して、ユーザー名を表示します。このオプションは、usernameEndString と一緒に使用されます。
usernameEndString	string	none	DN の最後から削除され、username を表示する文字列を定義します。このオプションは、usernameBeginString と一緒に使用されます。

オプション	Type	デフォルト	説明
roleNameAttributeID	attribute	name	ロール名を含む roleCtxDN コンテキスト内の属性の名前。roleAttributeIsDN プロパティーを true に設定すると、このプロパティーはロールオブジェクトの名前属性の検索に使用されます。
distinguishedNameAttribute	attribute	distinguishedName	ユーザーの DN を含むユーザーエントリーの属性の名前。これは、ユーザー自体の DN に特殊文字 (たとえば、正しいユーザーマッピングを防ぐバックスラッシュ) が含まれている場合に必要になることがあります。属性が存在しない場合は、エントリーの DN が使用されます。
roleRecursion	Integer	0	ロール検索の再帰レベルの数は、一致するコンテキストの下に続きます。これを 0 に設定して再帰を無効にします。
searchTimeLimit	integer	10000 (10 秒)	ユーザーまたはロールの検索のタイムアウト (ミリ秒単位)。
searchScope	OBJECT_SCOPE, ONELEVEL_SCOPE, SUBTREE_SCOPE のいずれか	SUBTREE_SCOPE	使用する検索範囲。
allowEmptyPasswords	true または false	false	空のパスワードを許可するかどうか。ほとんどの LDAP サーバーは、空のパスワードを匿名ログイン試行として処理します。空のパスワードを拒否するには、これを false に設定します。

オプション	Type	デフォルト	説明
referralUserAttributeIDToCheck	attribute	none	紹介を使用しない場合、このオプションは無視することができます。リファーラルを使用し、ロールオブジェクトがリファーラル内部にあると、このオプションは特定のロール (例: member) に対して定義されたユーザーが含まれる属性名を示します。ユーザーはこの属性名の内容に対して確認されます。このオプションが設定されていないとチェックは常に失敗するため、ロールオブジェクトはリファーラルツリーに保存できません。

認証は以下のように行われます。

1. LDAP サーバーへの最初のバインドは、**bindDN** オプションおよび **bindCredential** オプションを使用して行われます。**bindDN** は LDAP ユーザーであり、ユーザーとロールの **baseCtxDN** および **rolesCtxDN** ツリーの両方を検索する機能があります。認証するユーザー DN は、baseFilter 属性で指定されたフィルターを使用してクエリーされます。

2. 生成されるユーザー DN は、InitialLdapContext 環境 **Context.SECURITY_PRINCIPAL** として使用してユーザー DN を使用し、LDAP サーバーにバインドして認証されます。**Context.SECURITY_CREDENTIALS** プロパティーはコールバックハンドラーによって取得される String パスワードに設定されます。

4.1.1. LdapExtended ログインモジュールを使用するようセキュリティードメインを設定

データの例 (LDIF 形式)

```
dn: uid=jduke,ou=Users,dc=jboss,dc=org
objectClass: inetOrgPerson
objectClass: person
objectClass: top
cn: Java Duke
sn: duke
uid: jduke
userPassword: theduke
# ============================
dn: uid=hnelson,ou=Users,dc=jboss,dc=org
objectClass: inetOrgPerson
objectClass: person
objectClass: top
cn: Horatio Nelson
```

```
sn: Nelson
uid: hnelson
userPassword: secret
# ============================
dn: ou=groups,dc=jboss,dc=org
objectClass: top
objectClass: organizationalUnit
ou: groups
# ============================
dn: uid=ldap,ou=Users,dc=jboss,dc=org
objectClass: inetOrgPerson
objectClass: person
objectClass: top
cn: LDAP
sn: Service
uid: ldap
userPassword: randall
# ============================
dn: ou=Users,dc=jboss,dc=org
objectClass: top
objectClass: organizationalUnit
ou: Users
# ============================
dn: dc=jboss,dc=org
objectclass: top
objectclass: domain
dc: jboss
# ============================
dn: uid=GroupTwo,ou=groups,dc=jboss,dc=org
objectClass: top
objectClass: groupOfNames
objectClass: uidObject
cn: GroupTwo
member: uid=jduke,ou=Users,dc=jboss,dc=org
uid: GroupTwo
# ============================
dn: uid=GroupThree,ou=groups,dc=jboss,dc=org
objectClass: top
objectClass: groupOfUniqueNames
objectClass: uidObject
cn: GroupThree
uid: GroupThree
uniqueMember: uid=GroupOne,ou=groups,dc=jboss,dc=org
# ============================
dn: uid=HTTP,ou=Users,dc=jboss,dc=org
objectClass: inetOrgPerson
objectClass: person
objectClass: top
cn: HTTP
sn: Service
uid: HTTP
userPassword: httppwd
# ============================
dn: uid=GroupOne,ou=groups,dc=jboss,dc=org
objectClass: top
objectClass: groupOfUniqueNames
```

```
objectClass: uidObject
cn: GroupOne
uid: GroupOne
uniqueMember: uid=jduke,ou=Users,dc=jboss,dc=org
uniqueMember: uid=hnelson,ou=Users,dc=jboss,dc=org
```

LdapExtended ログインモジュールを追加する CLI コマンド

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

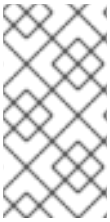
```
/subsystem=security/security-domain=testLdapExtendedExample/authentication=classic:add
```

```
/subsystem=security/security-domain=testLdapExtendedExample/authentication=classic/login-
module=LdapExtended:add( \
  code=LdapExtended, \
  flag=required, \
  module-options=[ \
    ("java.naming.factory.initial"=>"com.sun.jndi.ldap.LdapCtxFactory"), \
    ("java.naming.provider.url"=>"ldap://localhost:10389"), \
    ("java.naming.security.authentication"=>"simple"), \
    ("bindDN"=>"uid=ldap,ou=Users,dc=jboss,dc=org"), \
    ("bindCredential"=>"randall"), \
    ("baseCtxDN"=>"ou=Users,dc=jboss,dc=org"), \
    ("baseFilter"=>"(uid={0})"), \
    ("rolesCtxDN"=>"ou=groups,dc=jboss,dc=org"), \
    ("roleFilter"=>"(uniqueMember={1})"), \
    ("roleAttributeID"=>"uid"), \
  ])
```

```
reload
```

結果の XML

```
<security-domain name="testLdapExtendedExample" cache-type="default">
  <authentication>
    <login-module code="LdapExtended" flag="required">
      <module-option name="java.naming.factory.initial" value="com.sun.jndi.ldap.LdapCtxFactory"/>
      <module-option name="java.naming.provider.url" value="ldap://localhost:10389"/>
      <module-option name="java.naming.security.authentication" value="simple"/>
      <module-option name="bindDN" value="uid=ldap,ou=Users,dc=jboss,dc=org"/>
      <module-option name="bindCredential" value="randall"/>
      <module-option name="baseCtxDN" value="ou=Users,dc=jboss,dc=org"/>
      <module-option name="baseFilter" value="(uid={0})"/>
      <module-option name="rolesCtxDN" value="ou=groups,dc=jboss,dc=org"/>
      <module-option name="roleFilter" value="(uniqueMember={1})"/>
      <module-option name="roleAttributeID" value="uid"/>
    </login-module>
  </authentication>
</security-domain>
```



注記

上記の CLI コマンドは、JBoss EAP 6 のスタンドアロンインスタンスを想定して実行されました。JBoss EAP 6 ドメインでの CLI の使用の詳細は、Red Hat JBoss Enterprise Application Platform 6 Administration and Configuration Guide の **The Management CLI** セクションを参照してください。

4.1.1.1. Active Directory に LdapExtended ログインモジュールを使用するようセキュリティードメインを設定

Microsoft Active Directory では、LdapExtended ログインモジュールを使用できます。

デフォルトの AD 設定

以下の例は、デフォルトの Active Directory 設定を示しています。

Active Directory の設定によっては、通常のポート 389 ではなく、ポート 3268 でグローバルカタログを検索する必要がある場合があります。これは、Active Directory フォレストに複数のドメインが含まれる場合によく見られます。

デフォルト AD 設定用の LdapExtended ログインモジュールの設定例

```
<security-domain name="AD_Default" cache-type="default">
  <authentication>
    <login-module code="LdapExtended" flag="required">
      <module-option name="java.naming.provider.url" value="ldap://ldaphost.jboss.org"/>
      <module-option name="bindDN" value="JBOSSsearchuser"/>
      <module-option name="bindCredential" value="password"/>
      <module-option name="baseCtxDN" value="CN=Users,DC=jboss,DC=org"/>
      <module-option name="baseFilter" value="(sAMAccountName={0})"/>
      <module-option name="rolesCtxDN" value="CN=Users,DC=jboss,DC=org"/>
      <module-option name="roleFilter" value="(sAMAccountName={0})"/>
      <module-option name="roleAttributeID" value="memberOf"/>
      <module-option name="roleAttributeIsDN" value="true"/>
      <module-option name="roleNameAttributeID" value="cn"/>
      <module-option name="searchScope" value="ONELEVEL_SCOPE"/>
      <module-option name="allowEmptyPasswords" value="false"/>
    </login-module>
  </authentication>
</security-domain>
```

再帰的な AD 設定

以下の例では、Active Directory 内で再帰的なロール検索を実装します。この例と、デフォルトの Active Directory の例の主な違いは、ユーザーの DN を使用してメンバー属性を検索するためにロール検索が置き換えられている点です。ログインモジュールは、ロールの DN を使用して、グループがメンバーとなっているグループを検索します。

再帰検索を使用したデフォルトの AD 設定に対する LdapExtended ログインモジュールの設定例

```
<security-domain name="AD_Recursive" cache-type="default">
  <authentication>
    <login-module code="LdapExtended" flag="required">
      <module-option name="java.naming.provider.url" value="ldap://ldaphost.jboss.org"/>
      <module-option name="java.naming.referral" value="follow"/>
```

```
      <module-option name="bindDN" value="JBOSSsearchuser"/>
      <module-option name="bindCredential" value="password"/>
      <module-option name="baseCtxDN" value="CN=Users,DC=jboss,DC=org"/>
      <module-option name="baseFilter" value="(sAMAccountName={0})"/>
      <module-option name="rolesCtxDN" value="CN=Users,DC=jboss,DC=org"/>
      <module-option name="roleFilter" value="(member={1})"/>
      <module-option name="roleAttributeID" value="cn"/>
      <module-option name="roleAttributeIsDN" value="false"/>
      <module-option name="roleRecursion" value="2"/>
      <module-option name="searchScope" value="ONELEVEL_SCOPE"/>
      <module-option name="allowEmptyPasswords" value="false"/>
    </login-module>
  </authentication>
</security-domain>
```

第5章 データベースを使用するためのセキュリティードメインの設定

LDAP と同様に、セキュリティードメインはログインモジュールを使用して認証および承認にデータベースを使用するように設定できます。

5.1. DATABASE ログインモジュール

Database ログインモジュールは、認証とロールマッピングをサポートする JDBC (Java Database Connectivity) ベースのログインモジュールです。このログインモジュールは、ユーザー名、パスワード、およびロール情報がリレーショナルデータベースに格納される場合に使用されます。

このログインモジュールは、想定される形式のプリンシパルおよびロールが含まれる論理テーブルへの参照を提供して動作します。以下に例を示します。

```
Table Principals(PrincipalID text, Password text)
Table Roles(PrincipalID text, Role text, RoleGroup text)
```

Principals テーブルはユーザー PrincipalID を有効なパスワードに関連付けます。また、Roles テーブルはユーザー PrincipalID をそのロールセットに関連付けます。ユーザーパーミッションに使用されるロールは、Roles の RoleGroup コラムの値を持つ行に含まれる必要があります。

テーブルは論理であるため、ログインモジュールが使用する SQL クエリーをユーザーが指定できます。唯一の要件として、**java.sql.ResultSet** は前述の Principals および Roles と同じ論理構造を持ちます。テーブル名およびコラムの実際の名前は、コラムのインデックスに基づいてアクセスされるため、関係ありません。

この概念を明確化するために、すでに宣言されているようにPrincipals および Roles という 2 つのテーブルがあるデータベースを検討してください。以下のステートメントは、以下のデータをテーブルに追加します。

- Principals テーブルの **echoman** というパワスワードを持つ PrincipalID **java**
- Roles テーブルの **RolesRoleGroup** の **Echo** という名前のロールを持つ PrincipalID **java**
- Roles テーブルの **CallerPrincipalRoleGroup** に **caller-java** という名前のロールを持つ PrincipalID **java**

表5.1 完全なデータベースログインモジュールオプション

オプション	Type	デフォルト	説明
digestCallback	完全修飾クラス名	none	入力パスワードをハッシュするために salts などの事前/ ポストダイジェストコンテンツが含まれる DigestCallback 実装のクラス名。hashAlgorithm が指定されている場合にのみ使用されます。

オプション	Type	デフォルト	説明
dsJndiName	JNDI リソース	java:/DefaultDS	認証情報を格納している JNDI リソースの名前。このオプションは必須です。
hashAlgorithm	文字列	プレーンパスワードを使用	パスワードのハッシュに使用されるメッセージダイジェストアルゴリズム。サポートされているアルゴリズムは Java セキュリティープロバイダーによって異なりますが、サポートされているのは MD5、SHA-1、および SHA-256 です。
hashCharset	文字列	プラットフォームのデフォルトのエンコーディング	パスワード文字列をバイト配列に変換する際に使用する charset/エンコーディングの名前。これには、サポートされているすべての Java charset 名が含まれます。
hashEncoding	文字列	Base64	使用する文字列エンコーディング形式。
ignorePasswordCase	boolean	false	パスワードの比較で大文字と小文字を無視するかどうかを示すフラグ。
inputValidator	完全修飾クラス名	none	クライアントによって提供されるユーザー名およびパスワードを検証するために使用される InputValidator 実装のインスタンス。
principalsQuery	準備済み SQL ステートメント	select Password from Principals where PrincipalID=?	プリンシパルに関する情報を取得するための準備済み SQL クエリー。

オプション	Type	デフォルト	説明
rolesQuery	準備済み SQL ステートメント	none	ロールに関する情報を取得するための準備済み SQL クエリー。これは、「select Role, RoleGroup from Roles where PrincipalID=?」のクエリーと同等です。ここでは、Role は ロール名で、RoleGroup 列値は常に大文字の R または CallerPrincipal を持つ Roles のいずれかにしてください。
storeDigestCallback	完全修飾クラス名	none	入力パスワードをハッシュするために salts などのストア/予測ダイジェストコンテンツが含まれる DigestCallback 実装のクラス名。hashStorePassword または hashUserPassword が true で、hashAlgorithm が指定されている場合にのみ使用されます。
suspendResume	boolean	true	データベースの操作中に既存の JTA トランザクションを一時停止するかどうか。
throwValidatorError	boolean	false	検証エラーがクライアントへ公開されるべきかどうかを示すフラグ。
transactionManagerJndiName	JNDI リソース	java:/TransactionManager	ログインモジュールによって使用されるトランザクションマネージャーの JNDI 名。

5.1.1. データベースログインモジュールを使用するようセキュリティードメインを設定

Database ログインモジュールを使用するようにセキュリティードメインを設定する前に、データソースを適切に設定する必要があります。JBoss EAP 6 でのデータソースの作成と設定の詳細は、Red Hat JBoss Enterprise Application Platform 6 Administration and Configuration Guide のデータソース管理セクションを参照してください。

データソースを適切に設定したら、セキュリティードメインがデータベースログインモジュールを使用するように設定できます。以下の例では、**MyDatabaseDS** という名前のデータソースが作成され、以下で構築されるデータベースで正しく設定されていることを前提としています。

```
CREATE TABLE Users(username VARCHAR(64) PRIMARY KEY, passwd VARCHAR(64))
CREATE TABLE UserRoles(username VARCHAR(64), role VARCHAR(32))
```

データベースログインモジュールを追加する CLI コマンド

```
/subsystem=security/security-domain=testDB:add
```

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

```
/subsystem=security/security-domain=testDB/authentication=classic/login-module=Database:add( \
  code=Database, \
  flag=required, \
  module-options=[ \
    ("dsJndiName"=>"java:/MyDatabaseDS"), \
    ("principalsQuery"=>"select passwd from Users where username=?"), \
    ("rolesQuery"=>"select role, 'Roles' from UserRoles where username=?") \
  ])
```

```
reload
```

結果の XML

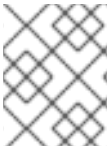
```
<security-domain name="testDB">
  <authentication>
    <login-module code="Database" flag="required">
      <module-option name="dsJndiName" value="java:/MyDatabaseDS"/>
      <module-option name="principalsQuery" value="select passwd from Users where username=?"/>
      <module-option name="rolesQuery" value="select role, 'Roles' from UserRoles where
username=?"/>
    </login-module>
  </authentication>
</security-domain>
```

第6章 ファイルシステムを使用するためのセキュリティードメインの設定

セキュリティードメインは、ログインモジュールを使用して認証および承認のアイデンティティーストアとしてファイルシステムを使用するように設定することもできます。

6.1. USERSROLES ログインモジュール

UsersRoles ログインモジュールは、Java プロパティーファイルからロードされる複数のユーザーおよびユーザーロールをサポートする簡単なログインモジュールです。このログインモジュールの主な目的は、アプリケーションとともにデプロイされたプロパティーファイルを使用して複数のユーザーおよびロールのセキュリティー設定を簡単にテストすることです。デフォルトの username-to-password マッピングファイル名は **users.properties** で、デフォルトの username-to-roles マッピングファイル名は **roles.properties**です。



注記

このログインモジュールは、パスワードスタッキング、パスワードハッシュ、および認証されていないアイデンティティーをサポートします。

プロパティーファイルは、initialize メソッドスレッドコンテキストローダーを使用して初期化中にロードされます。つまり、これらのファイルは Jakarta EE デプロイメントのクラスパス (WAR アーカイブの **WEB-INF/classes** フォルダーなど) またはサーバークラスパスの任意のディレクトリーに配置できます。

表6.1 完全な UsersRoles ログインモジュールオプション

オプション	Type	デフォルト	説明
usersProperties	ファイルまたはリソースへのパス。	users.properties	ユーザー/パスワード間のマッピングが含まれるファイルまたはリソースです。ファイルの形式は username=password です。
rolesProperties	ファイルまたはリソースへのパス。	roles.properties	ユーザー/ ロール間のマッピングが含まれるファイルまたはリソースです。ファイルの形式は username=role1,role2,role3 です。

オプション	Type	デフォルト	説明
password-stacking	useFirstPass または false	false	useFirstPass の値は、このログインモジュールが最初に LoginContext に格納されている情報を調べて ID を探す必要があることを示します。このオプションは、このログインモジュールと他のログインモジュールをスタックする際に使用できます。
hashAlgorithm	パスワードハッシュアルゴリズムを表す文字列。	none	パスワードをハッシュするために使用される java.security.MessageDigest アルゴリズムの名前。デフォルトはないため、ハッシュを有効にするには、このオプションを明示的に設定する必要があります。hashAlgorithm が指定されている場合、CallbackHandler から取得されたクリアテキストパスワードは、inputPassword 引数として UsernamePasswordLoginModule.validatePassword に渡される前にハッシュされます。users.properties ファイルに保存されているパスワードは、同様にハッシュ化する必要があります。
hashEncoding	base64 または hex	base64	hashAlgorithm も設定されている場合、ハッシュ化されたパスワードの文字列形式。
hashCharset	string	コンテナーのランタイム環境に設定されるデフォルトのエンコーディング	クリアテキストのパスワードをバイトアレイに変換するために使用されるエンコーディング。

オプション	Type	デフォルト	説明
unauthenticatedIdentity	プリンシパル名	none	認証情報を含まないリクエストに割り当てられるプリンシパル名を定義します。これを使用すると、保護されていないサーブレットは特定ロールを必要としない EJB でメソッドを呼び出すことができます。このようなプリンシパルには関連したロールがなく、セキュアでない EJB や、チェックされていないパーミッション制約と関連する EJB メソッドのみにアクセスできます。

6.1.1. UsersRoles ログインモジュールを使用するようにセキュリティードメインを設定

以下の例では、以下のファイルが作成され、アプリケーションのクラスパスで利用できることを前提としています。

- sampleapp-users.properties
- sampleapp-roles.properties

UserRoles ログインモジュールを追加する CLI コマンド

```
/subsystem=security/security-domain=sampleapp:add
```

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

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

```
reload
```

結果の XML

```
<security-domain name="sampleapp">
  <authentication>
    <login-module code="UsersRoles" flag="required">
      <module-option name="usersProperties" value="sampleapp-users.properties"/>
      <module-option name="rolesProperties" value="sampleapp-roles.properties"/>
```

```
    </login-module>
  </authentication>
</security-domain>
```

第7章 セキュリティードメインがセキュリティーマッピングを使用するように設定

セキュリティードメインにセキュリティーマッピングを追加すると、認証または承認の実行後、情報がアプリケーションに渡される前に認証および承認情報を組み合わせることができます。セキュリティーマッピングの詳細は、Security Mapping section of the Red Hat JBoss Enterprise Application Platform 6 Security Architecture guide を参照してください。

既存のセキュリティードメインにセキュリティーマッピングを追加するには、**code**、**type**、および関連するモジュールオプションを設定する必要があります。**code** フィールドは、セキュリティーマッピングモジュールの短い名前 (**SimpleRoles**、**PropertiesRoles**、**DatabaseRoles など**) またはクラス名です。**type** フィールドは、このモジュールが実行するマッピングのタイプを指し、許可される値は **principal**、**role**、**attribute**、または **credential** です。使用可能なセキュリティーマッピングモジュールとそのモジュールオプションの完全なリストについて は、Red Hat JBoss Enterprise Application Platform 6 セキュリティーガイドのセキュリティーマッピングモジュールのセクションを 参照してください。

例: SimpleRoles セキュリティーマッピングを既存のセキュリティードメインに追加する CLI コマンド

```
/subsystem=security/security-domain=sampleapp/mapping=classic:add
```

```
/subsystem=security/security-domain=sampleapp/mapping=classic/mapping-
module=SimpleRoles:add( \
code=SimpleRoles, \
type=role, \
module-options=[("user1"=>"specialRole")])
```

```
reload
```

結果の XML

```
<security-domain name="sampleapp">
  <authentication>
  ...
  </authentication>
  <mapping>
    <mapping-module code="SimpleRoles" type="role">
      <module-option name="user1" value="specialRole"/>
    </mapping-module>
  </mapping>
</security-domain>
```

第8章 スタンドアロン VS。ドメインモードに関する考慮事項

LDAP サーバー (Microsoft Active Directory を含む) またはその他の ID ストアを使用して、セキュリティーレルムとセキュリティードメインの両方の認証および/または承認のためにアイデンティティー管理を設定することは、スタンドアロンドメインと管理ドメインのどちらで使用されているかに関係なく基本的に同じです。他の設定と同様に、スタンドアロン設定は **standalone.xml** ファイルにあり、管理対象ドメインの設定は **domain.xml** および **host.xml** ファイルにあります。