App Connect Enterprise
資格情報管理

- 第2.0版（2024.1.12）

日本アイ・ビー・エム システムズ・エンジニアリング

はじめに

- 本書は、IBM App Connect Enterprise(ACE) で提供するVaultやServerCredentialsについて記載したものです

- 本書の内容はACE V12 Fixpack 10を使用した検証を元に執筆しています。
  - 将来のバージョンアップによってコマンド等が変わる可能性があります。

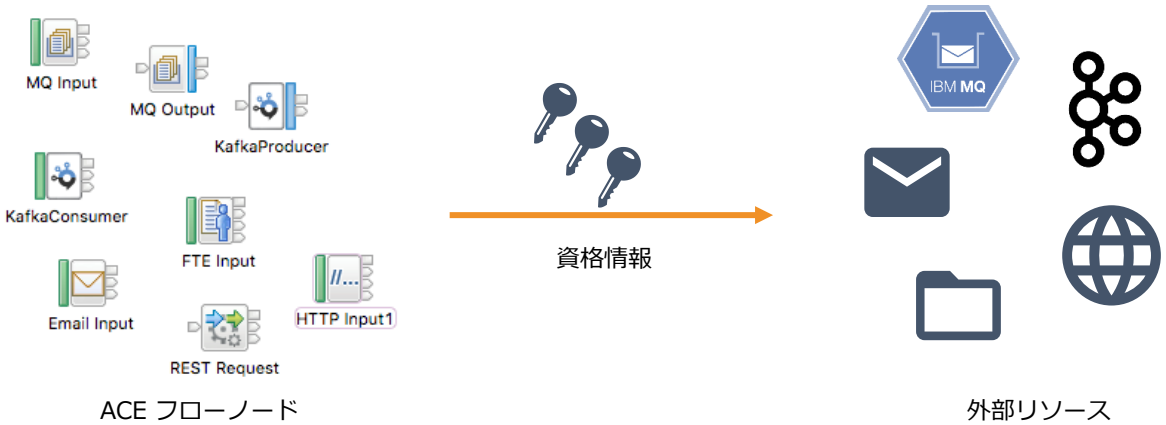
内容

- ACEにおける資格情報管理
  - ACEと資格情報
  - ACEにおける資格情報管理
    - Vault
    - ServerCredentials
    - 管理方法
- Vaultの構成
  - 独立統合サーバーでの構成
  - .mqsivaultrc
  - 統合ノードでの構成
  - 外部ディレクトリでの構成
  - Connector Discovery Wizardを使用したVaultの構成
  - 外部ソースからの資格情報の取得
  - 資格情報の登録・更新・削除
  - 資格情報のインポート/エクスポート
  - Java Computeからの資格情報へのアクセス
- ServerCredentialsの構成
- REST APIとWeb UI
- Vault保管の認証情報を使用したBasic認証

ACEにおける資格情報の管理

ACEと資格情報

- App Connect Enterprise(ACE)では様々な外部リソースとの接続を行い、システム間連携/インテグレーションを実現する
  - データベース、MQ、Kafka、FTP、REST API ...
- 外部リソースとの接続には資格情報(e.g. ユーザーID/パスワード)が必要となる
- 資格情報をACEで一元管理する仕組みとして以下を提供
  - mqsisetdbparms
  - Vault
  - Server Credentials

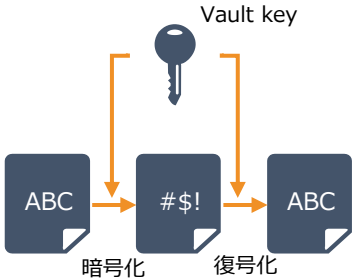


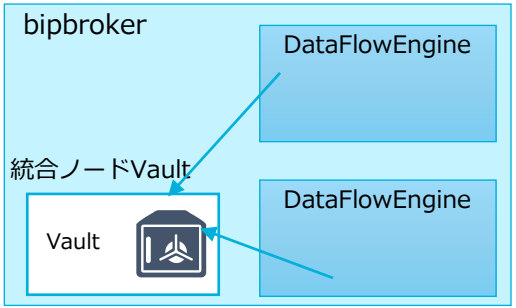
ACEにおける資格情報管理

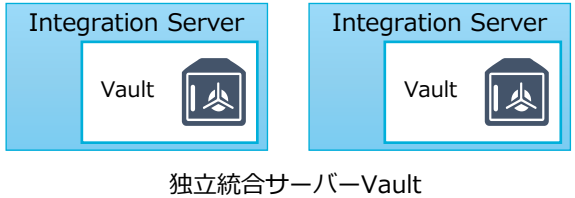
- mqsisetdbparmsコマンド
  - IIBまでの資格情報管理の方法
  - 資格情報は製品独自のアルゴリズムで変換してファイルに書き込まれる
  - 資格情報ごとにディレクトリ/ファイルを生成する
    - ファイルの格納先
      - ✓ 統合ノード構成: /var/mqsi/registry/<Integration_node_name>/CurrentVersion/DSN
      - ✓ 独立統合サーバー構成: <work_dir>/config/registry/integration_server/CurrentVersion/DSN
- Vault
  - ACE V11.0.0.6以降に出てきた資格情報管理の方法
  - 暗号化ストアを使用することで、mqsisetdbparmsよりセキュアに資格情報の管理が可能
- Server Credentials
  - ACEのコンテナ化に対応し、コンテナ/OpenShift環境で容易に資格情報管理を実現
- 基本的な機能はどの管理方法でも同等
  - 一部、Vaultのみ構成可能な機能(Vaultを用いたBasic認証、後述)もある
  - 既存の環境からの移行の場合にはmqsisetdbparmsを継続利用する、コンテナ環境ではServerCredentialsを使うなど、用途によって使い分けや併用も可能

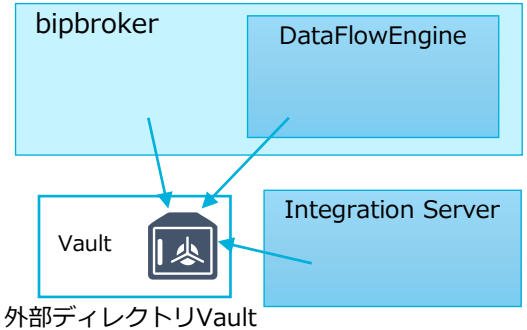
Vault

- ランタイム上で資格情報を管理するコンポーネントとして新たに登場
  - セキュアな暗号化ストアを提供
    - Vault内のデータは対称鍵暗号方式で暗号化/復号化を行う
    - Vaultの作成や削除、キーの変更などの操作はmqsivaultコマンドで行う
  - 統合ノードVault
    - V12.0.9.0以降、統合ノード、統合ノードリスナー、統合ノード下の統合サーバから利用可能なVault
      - ✓ 特定の統合ノード配下のコンポーネント間で資格情報の共有・一元管理が可能
      - ✓ V12.0.8.0までは統合サーバは各々Vaultを保持
  - 独立統合サーバーVault
    - 特定の統合サーバからのみ利用されるVault
    - 独立統合サーバーのワークディレクトリに構成
  - 外部ディレクトリVault
    - 複数の統合サーバ、統合ノードで共有可能なVault









ServerCredentials

- server.conf.yaml上で資格情報を定義することが可能
  - 独立統合サーバー環境でのみ使用可能
- コンテナでのユースケースを想定
  - KubernetesやOpenShift環境では、server.conf.yamlをSecretやConfigMapリソースとして定義し、コンテナ起動時に読み込ませて構成することが可能

server.conf.yaml

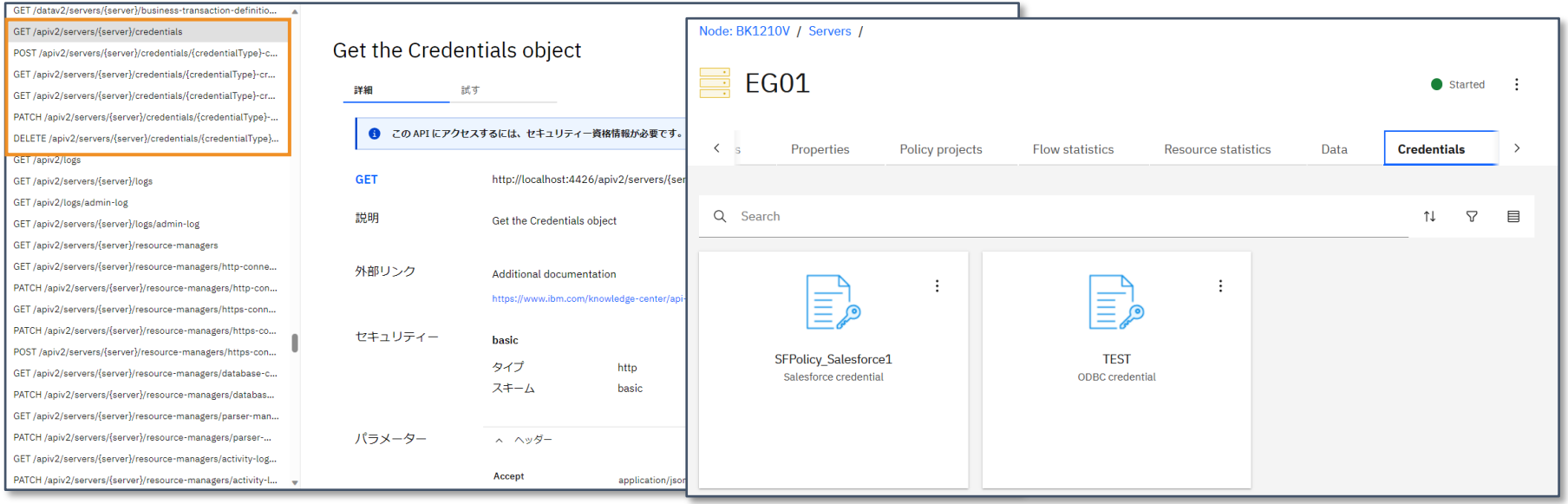
```
ServerCredentials:
  http:
    SecProfID:
      password: "password"
      username: "joe"
    SecProfID:
     password: "drowssap"
      username: "alice"
  rest:
    remoteApiID1:
      username: "RestUser"
      password: "RestPwd"
    remoteApiID2:
      apiKey: "RestApiKey"
```

ace-pod.yaml

```
kind: DeploymentConfig
apiVersion: apps.openshift.io/v1
spec:
(- snip -)
  template:
    spec:
      containers:
        - name: aceserver1
          image: ibm/ibmace:latest
          volumeMounts:
            - name: serverconf
              mountPath: "/home/aceuser/initial-config/serverconf"
      volumes:
        - name: serverconf
          secret:
            secretName: aceserver1-conf-yaml
            items:
            - key: serverconf
              path: server.conf.yaml
```

管理方法

- 管理REST API
  - 資格情報の作成(POST)と定義された資格情報の確認(GET)が可能
  - 一部の資格情報では更新（PATCH)、削除（DELETE)も可能
- web UI
  - credentialsのタブから、資格情報の一覧と、作成タイプ(mqsisetdbparms/vault/ServerCredentials)が確認できる



Vaultの構成

独立統合サーバーでの構成

- Valutの作成(mqsivaultコマンド)
  - 独立統合サーバーの稼働中に作成できないため、既存の環境に作成する場合は停止しておく
    - コマンドラインでVault keyを直接指定して構成する場合

```
mqsivault --work-dir /home/aceuser/ace-server/ --create --vault-key <my_vaultkey>
```

    - .mqsivaultrc(Valut keyを管理するファイル)を使用して構成する場合(.mqsivaultrcの詳細はP.16参照)
      - ✓ .mqsivaultrcを使用することで、Vault keyが必要なコマンドを実行する度にキーそのものを入力する代わりにファイルパスを指定するだけで済む

```
# あらかじめ.mqsivaultrcへVault keyを登録しておく
mqsivault --work-dir /home/aceuser/ace-server/ --vaultrc-store-key ¥
--vault-key <my_vaultkey> --vaultrc-location /home/aceuser/

# Vaultの作成
mqsivault --work-dir /home/aceuser/ace-server --create --vaultrc-location /home/aceuser/
```

  - Tips
    - Vault keyは8文字以上の文字列である必要あり
    - Vault keyが流出すると、資格情報の更新や削除ができる権限を与えることになるため、取り扱いは注意が必要
      - ✓ mqsicredentialsコマンドのreportオプションでは、資格情報名やタイプ、ユーザーIDは取得できるが、パスワードやAPIキーといった情報は取得できない

独立統合サーバーでの構成

- Vaultを構成した独立統合サーバー起動
  - IntegrationServerコマンドでの起動時にVault keyか.mqsivaultrcのパスを指定
    - IntegrationServer --work-dir <dir> --vault-key <key>
    - IntegrationServer --work-dir <dir> --vaultrc-location <dir>
  - 起動時に--vault-key または --vaultrc-locationでVault keyを渡さない場合は起動に失敗する

参考: 独立統合サーバー起動時のエラー

- 誤ったVault keyで起動

```
[aceadmin@ISEP20220119-1458-mysterious-01 work]$ IntegrationServer --work-dir /work/ace-server --
vault-key Wrongkey
2023-11-30 14:54:25.786580: BIP1990I: 統合サーバー 'ace-server' は初期化を開始しています。バージョン
'12.0.10.0' (64 ビット)。
main checker destroyed before initialisation completed, latest stage was Creating credentials
manager , pid 288022, tid 288022, attempted stage was 9 and 8 stages completed
2023-11-30 14:54:25.864416: BIP2117E: IBM App Connect Enterprise の内部エラー。 診断情報 'Fatal
Error; exception thrown before initialisation completed'、'Creating credentials manager'、''、
'288022'、'288022'、'9'、'8'。
2023-11-30 14:54:25.864496: BIP2203E: 統合サーバーが、開始中に問題を検出しました。
2023-11-30 14:54:25.864560: BIP7219E: 提供されたパスフレーズを使用して、ロケーション '/work/ace-
server/config/vault' にある暗号化ストアを開くことができませんで した。
2023-11-30 14:54:25.864606: BIP7229E: 指定されたパスフレーズを使用してバッファーを暗号化解除できません。
2023-11-30 14:54:25.864690: BIP1992I: 統合サーバー 'ace-server' が停止しました。
```

.mqsivaultrc

- 独立統合サーバーや統合ノードのVault keyを管理するファイル
  - 複数の独立統合サーバー/統合ノードのキーを管理可能
    - ワークディレクトリ(独立統合サーバーの場合)、または統合ノード名に続いてVault keyがbase64エンコーディングされた状態で保存される
    - サーバー起動時や、mqsicredentialコマンド等、Vault keyが必要なコマンド実行の度にキーを入力する必要が不要
      - ✓ スクリプト等でサーバーを起動する場合に、Vault keyをスクリプト内に記述する必要がなくなる
  - ファイルのパーミッションは600(所有者のみ読み書きが可能)で作成される
    - .msiqvaultrcに格納されたVault keyはbase64でエンコードされているだけなので、容易にデコードが可能
      - ✓ ファイルのパーミッションの扱いは注意が必要

```
$ cat .mqsivaultrc
Server=/home/aceuser/work/SIS4=bXl2YXVsdGtleQ==
Node=ACEBROKER2=bXl2YXVsdGtleQ==
```

- 参照の優先順位
  - コマンドの --vaultrc-location オプション指定の他に、以下の順番で.mqsivaultrcを参照する
    - --vaultrc-locationオプション
    - 環境変数: MQSI_VAULTRC_LOCATION
    - 環境変数: HOME/HOMEPATH

統合ノードでの構成

- Vaultの作成(mqsivaultコマンド)
  - 統合ノード作成時にVaultも作成する
    - コマンドラインでVault keyを直接指定して構成する場合

```
mqsicreatebroker ACEBROKER1 --vault-key <my_vaultkey>
```

    - .mqsivalultrcを使用する場合

```
# あらかじめ.mqsivaultrcへVault keyを登録しておく
mqsivault --vaultrc-store-key --vault-key myvaultkey --vaultrc-location ¥
/home/aceuser/ ACEBROKER2
mqsicreatebroker ACEBROKER2 --vaultrc-location <dir>
```

  - 統合ノード作成後に作成する(統合ノードは停止しておく必要がある)
    - コマンドラインでVault keyを直接指定して構成する場合

```
mqsivault ACEBROKER3 --create --vault-key myvaultkey
```

    - .mqsivalultrcを使用する場合

```
mqsivault ACEBROKER4 --vaultrc-store-key --vault-key myvaultkey --vaultrc-location ¥
/home/aceuser/
mqsivault ACEBROKER4 --create --vaultrc-location <dir>
```

統合ノードでの構成

- 統合ノードの起動
  - Vault key指定でVaultを作成した場合は起動時にもVault keyを指定する

```
mqsistart ACEBROKER1 --vault-key <my_vaultkey>
```

  - .mqsivaultrcを使用するように構成した場合はファイルのパスを指定する

```
mqsistart ACEBROKER2 --vaultrc-location /home/aceuser/
```

- 統合サーバーでVaultを使用する
  - Vaultが構成された統合ノード下で稼働する統合サーバーも作成されたVaultにアクセスが可能
    - 統合ノードのVault keyが統合サーバにも共有される

参考: 統合ノード作成時にVaultを作成

- --vault-key オプションでキーを渡す方法

```
# 統合ノードの作成
$ mqsicreatebroker ACEBK1 --vault-key myvaultkey
BIP8071I: コマンドは正常終了しました。
# Vault key無しでは起動不可（デフォルトでホーム・ディレクトリを参照しにいく）
$ mqsistart ACEBK1
BIP8873I: コンポーネント 'ACEBK1' のコンポーネント検査を開始します。
BIP15162I: デフォルトとして、ユーザーのホーム・ディレクトリー内でボールト rc フ ァイルを検索します。
BIP15163I: ボールト rc ファイル '/home/aceadmin/.mqsivaultrc' から取得されたボールト・キー。
BIP15152E: 指定されたキーは、ロケーション '/var/mqsi/components/ACEBK1/config/vault' のボールトのアンロックに使用できません。
指定されたキーでは暗号化されたストアをアンロックできなかったため、 ロケーション
'/var/mqsi/components/ACEBK1/config/vault' のボールトにアクセスする要求は失敗し ました。
正しいボールト・キーを使用してコマンドを再試行してください。
# Vault keyを渡して起動する
$ mqsistart ACEBK1 --vault-key myvaultkey
BIP8873I: コンポーネント 'ACEBK1' のコンポーネント検査を開始します。
BIP15293I: Node or workdir vault key obtained from --vault-key command line argument.
BIP8096I: コマンドが正常に開始されました。システム・ログを調べて、コンポーネントが問題なく開始されたこと、および、実行が問題なく継続されていることを確認してください。
```

参考: 統合ノード作成後にVaultを作成

- --vault-key オプションでキーを渡す方法

```
# 統合ノードの作成
$ mqsicreatebroker ACEBK1
BIP8071I: コマンドは正常終了しました。
# 統合ノード ACEBK1に対するVaultの作成
$ mqsivault ACEBK1 --create --vault-key myvaultkey
BIP15293I: Node or workdir vault key obtained from --vault-key command line argument.
BIP8071I: コマンドは正常終了しました。
# 統合ノード起動時にはVault作成時に設定したVault keyを指定する
$ mqsistart ACEBK1 --vault-key myvaultkey
BIP8873I: コンポーネント 'ACEBK1' のコンポーネント検査を開始します。
BIP15293I: Node or workdir vault key obtained from --vault-key command line argument.
BIP8096I: コマンドが正常に開始されました。システム・ログを調べて、コンポーネントが問題なく開始されたこと、および、実行が問題なく継続されていることを確認してください。
```

参考: 統合ノード作成後にVaultを作成

- --vaultrc-locationで.mqsivaultrcファイルの場所を指定する方法

```
# .mqsivaultrcのロケーションを指定して統合ノードを作成
$ mqsicreatebroker ACEBK2 --vaultrc-location /home/aceadmin
BIP15296I: Using the .mqsivaultrc file in the directory explicitly passed in.
BIP8071I: コマンドは正常終了しました。
# Vaultを作成
$ mqsivault ACEBK2 --vaultrc-store-key --vault-key myvaultkey
BIP15170I: ユーザーのホーム・ディレクトリー '/home/aceadmin' 内のボールト rc フ ァイルを更新しています。
BIP8071I: コマンドは正常終了しました。
# mqsivaultコマンドで.mqsivaultrcファイルが作成される
$ cat .mqsivaultrc
Node=ACEBK2=bXl2YXVsdGtleQ==
# ユーザーのホームディテクトリに.mqsivaulrcを作成した場合は、--vaultrc-location無しでも起動する
$ mqsistart ACEBK2
BIP8873I: コンポーネント 'ACEBK2' のコンポーネント検査を開始します。
BIP8096I: コマンドが正常に開始されました。システム・ログを調べて、コンポーネントが問題なく開始されたこと、および、実行が問題なく継続されていることを確認してください。
```

外部ディレクトリでの構成

- Vaultの作成(mqsivaultコマンド)
  - 外部ディレクトリにVaultを作成する
    - コマンドラインでVault keyを直接指定して構成する場合

```
mqsivault –-ext-vault-dir <extvault_dir> --create –-ext-vault-key <ext_my_vaultkey>
```

    - .mqsivaultrcを使用する場合

```
# あらかじめ.mqsivaultrcへ外部ディレクトリVault keyを登録しておく
mqsivault --vaultrc-store-ext-key –-ext-vault-dir <extvault_dir> –-ext-vault-key ¥
<ext_my_vaultkey> --vaultrc-location <dir>
# Vaultの作成
mqsivault --ext-vault-dir <extvault_dir> --create --ext-vault-key <ext_my_vaultkey>
```

参考: 統合サーバーで外部ディレクトリVaultを使用

- --ext-vault-keyオプションでキーを渡す方法

```
# 統合サーバでワークディレクトリを作成
$ mqsicreateworkdir IS1
mqsicreateworkdir: Copying sample server.config.yaml to work directory
1 file(s) copied.
コマンドは正常終了しました。
# server.conf.yamlを編集
   （directoryに外部ディレクトリVaultのPathを記載）
   Credentials:
     ExternalDirectoryVault:
       directory: '/work/vault_test/extdir'
# 外部 Vault keyを渡して起動する
$ IntegrationServer --work-dir /work/IS1 --ext-vault-key extmykey
2023-12-01 16:50:46.652184: BIP1990I: 統合サーバー 'IS1' は初期化を開始していま す。バージョン '12.0.10.0' (64
ビット)。
2023-12-01 16:50:46.734788: BIP10104I: Using external directory vault '/work/vault_test/extdir'.
2023-12-01 16:50:46.741476: BIP9905I: リソース・マネージャーを初期化中です。
2023-12-01 16:50:47.513584: BIP9398I: 「mqsicdc.par」からのリソースがロードされ ていません。
2023-12-01 16:50:50.961208: BIP9906I: デプロイ済みのリソースを読み取り中です。
2023-12-01 16:50:51.508272: BIP2866I: IBM App Connect Enterprise 管理セキュリテ ィーは inactive です。
2023-12-01 16:50:51.524732: BIP3132I: HTTP リスナーがポート '7600' で 'RestAdmin http' 接続の listen を開始しま
した。
2023-12-01 16:50:51.526028: BIP1991I: 統合サーバーが初期化を完了しました。
```

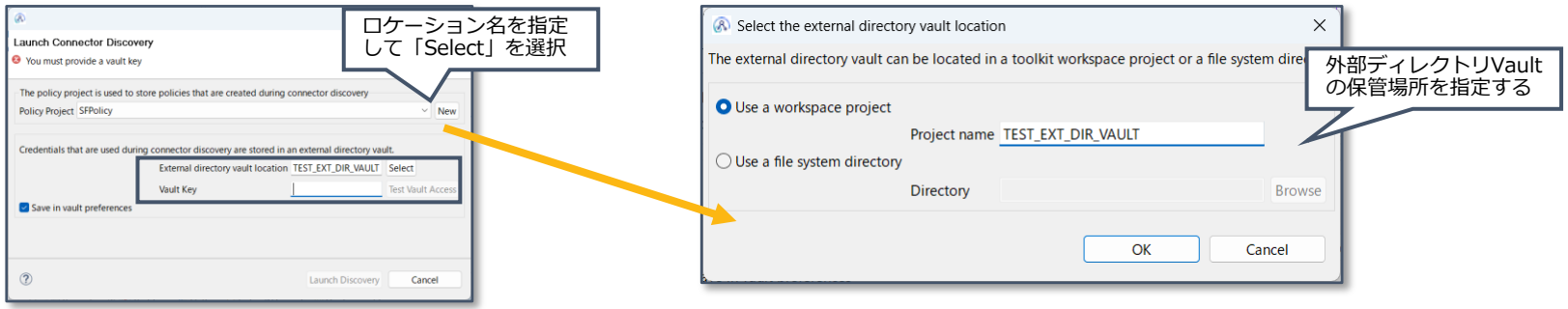
参考: 統合ノードで外部ディレクトリVaultを使用

- --vaultrc-locationで.mqsivaultrcファイルの場所を指定する
  - mqsicreatebrokerコマンドでは--ext-vault-keyオプションがないため、統合ノード作成時に外部ディレクトリVaultを構成するには.mqsivaultrcファイルの指定が必要

```
# Vaultと.mqsivaultrcファイルを作成
$ mqsivault --vaultrc-store-ext-key --ext-vault-dir /work/vault_test/extdir2 --ext-vault-key ext2mykey --va
ultrc-location /work/vault_test
BIP8071I: コマンドは正常終了しました。
$ mqsivault --ext-vault-dir /work/vault_test/extdir2 --create --ext-vault-key ext2mykey
BIP15294I: The external directory vault key has been obtained from the --ext-vault-key command line parameter.
BIP8071I: コマンドは正常終了しました。
#.mqsivaultrcファイルが作成される
$ cat .mqsivaultrc
ExtDir=/work/vault_test/extdir2=ZXh0Mm15a2V5
# .mqsivaultrcのロケーションを指定して統合ノードを作成＆起動
$ mqsicreatebroker ACEBK3 --ext-vault-dir /work/vault_test/extdir2 --vaultrc-location /work/vault_test
BIP15296I: Using the .mqsivaultrc file in the directory explicitly passed in.
BIP15165I: ボールト・キーがボールト rc ファイル '/work/vault_test/.mqsivaultrc' で見つかりません。
BIP8071I: コマンドは正常終了しました。
$ mqsistart ACEBK3 --vaultrc-location /work/vault_test
BIP8873I: コンポーネント 'ACEBK3' のコンポーネント検査を開始します。
BIP15296I: Using the .mqsivaultrc file in the directory explicitly passed in.
BIP15297I: The external directory vault key has been obtained from the vault rc file
'/work/vault_test/.mqsivaultrc'.
BIP8096I: コマンドが正常に開始されました。システム・ログを調べて、コンポーネントが問題なく開始されたこと、および、実行が問題なく継続されていることを確認してください。
```

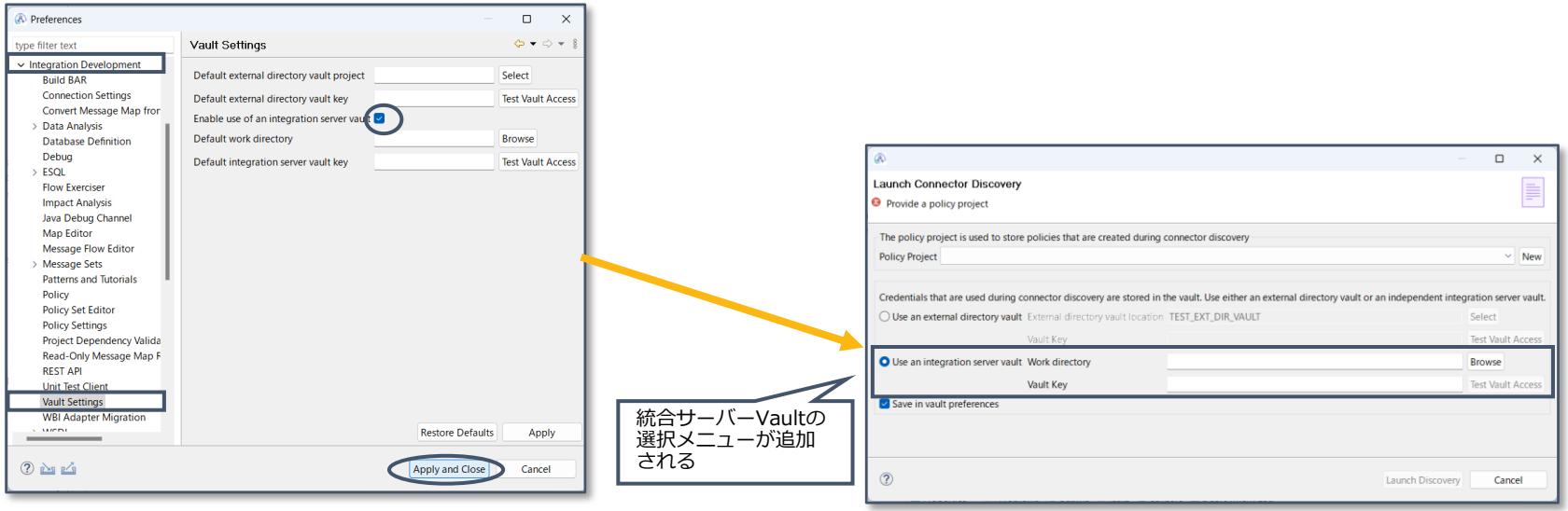
Connector Discovery Wizardを使用したVaultの構成

- Connector Discovery Wizardを使用したVaultの構成
  - Connector Discovery Wizardを使用したノード構成時に、エンドポイント・アプリケーションに接続するための資格情報を保管するためにVaultを指定
    - 指定時にVaultが作成されていなければ作成
  - デフォルトは外部ディレクトリー・ボルトに保管
    - 独立統合サーバのボルトを使用する場合は、事前にツールキットの設定が必要（次ページ）
    - 統合ノードのボルトは指定不可



Connector Discovery Wizardを使用したVaultの構成

- 独立統合サーバーVaultを使用する場合は事前に設定が必要
  - Toolkitで「Window」＞「Preferences」＞「Integration Development」＞「Vault Settings」で設定
    - ✓ 「Enable use of an integration server vault」を選択し、「Apply and Close」で設定



外部ソースからの資格情報の取得

- V12.0.3.0以降、外部ソースから資格情報を取得することも可能
  - 統合サーバの起動時にコマンドを実行し、外部ソースから取得した資格情報を統合サーバに取り込むことが可能
    - 任意の外部のVaultサーバーとの連携が可能
  - 読み込めるフォーマットは規定されているため、詳細はマニュアルを参照
    https://www.ibm.com/docs/en/app-connect/12.0?topic=cis-configuring-integration-server-use-security-credentials-from-external-source
  - server.conf.yamlのExternalCredentialsProvidersのエントリで設定

```
ExternalCredentialsProviders:
  MyExternalCredentialProvider1:                    ・・・①
      loadAllCredentialsCommand: '/home/aceuser1/ace-server1/read-external-credentials.sh'   ・・・②
      loadAllCredentialsFormat: 'xml'               ・・・③
      loadAllCredentialsCodepage: 1208              ・・・④
      loadAllCredentialsDirectErrorToCommandOutput: true    ・・・⑤
      loadAllCredentialsIncludeCommandOutputInLogs: false   ・・・⑥
```

① 外部資格情報プロバイダー名（ユニークな名前）
② ロードする資格情報を取得するために実行するスクリプトなどを指定
③ コマンドによって返される出力のフォーマットを指定
　指定可能な値は、xml、json、yamlのいずれか
④ コマンドによって返される出力のコード・ページを指定
⑤⑥はデバッグ目的でのみ使用（本番環境では設定しない）
　⑤ stderr出力をstdoutストリームにリダイレクトするかどうか
　⑥ 資格情報をログに出力するかどうか

資格情報の登録・更新・削除

- mqsicredentialコマンド
  - 資格情報を暗号化してVaultに登録・参照・更新・削除を行うコマンド
    - サーバー起動中の更新・削除は不可
  - 資格情報の型によって指定できるオプションは異なるので、詳細はマニュアルを参照
    マニュアル: https://www.ibm.com/docs/ja/app-connect/12.0?topic=commands-mqsicredentials-command

- コマンド体系

```
mqsicredentials
--create | --report |--update | --delete # 作成、参照、更新、削除を指定
--workdir <work_dir>| <integration_node_name> -e <integration_server_name>
  | --ext-vault-dir <ext_dir>
  # 独立統合サーバーのworkディレクトリまたは、統合ノード、統合サーバー名または、外部ディレクトリ
--credential-type <type> # 資格情報のタイプを指定
--credential-name <cred_name> # 資格情報名(データソース名)を指定
<credentials_options> # 資格情報のタイプによって異なる
--vault-key | --ext-vault-key | --vaultrc-location
  # Vault keyまたは、外部ディレクトリVault keyまたは、.mqsivaultrcのディレクトリを指定
```

参考: コマンド実行例

- 統合ノードにODBCの資格情報を定義

```
# 統合ノードACEBK1のVaultに対してODBCの資格情報を定義
C:¥Program Files¥IBM¥ACE¥12.0.10.0>mqsicredentials --create ACEBK1 -e EG01 --credential-
type odbc --credential-name TEST --username db2inst1 --password db2inst1
BIP15119I: タイプ 'odbc' の資格情報名 'TEST' の 'create' アクションは正常に終了しました。

BIP8071I: コマンドは正常終了しました。

# 作成した資格情報を参照
# 登録した統合ノードVaultと、参照している外部ディレクトリVaultの情報がみれる
C:¥Program Files¥IBM¥ACE¥12.0.10.0>mqsicredentials --report ACEBK1 -e EG01
BIP15110I: タイプ 'odbc' の資格情報名 'TEST' には、プロバイダー 'servervault' のユーザー名
'db2admin' が含まれており、以下のプロパティーが定義されています: 'password'
BIP15121I: プロバイダー 'extdirvault' のタイプ 'salesforce' の資格情報名
'SFPolicy_Salesforce1' には、以下のプロパティーが定義されています: 'clientId, clientSecret,
accessToken, refreshToken'

BIP8071I: コマンドは正常終了しました。
```

資格情報のインポート/エクスポート

- V12.0.10.0でVaultの中身をインポート/エクスポート可能に
  - mqsivaultの--export オプション、--import オプションを使用
    - --export オプションでvaultの中身をアーカイブ・ファイルに出力(zip形式)
      - ✓ 中身はアーカイブ・キーで暗号化
    - --import オプションでアーカイブ・ファイルの中身を宛先のVaultにインポート

- コマンド体系

```
mqsivault
--work-dir <work_dir>| <integration_node_name> -e <integration_server_name>
  | --ext-vault-dir <ext_dir>
  # 独立統合サーバーのworkディレクトリまたは、統合ノード、統合サーバー名または、外部ディレクトリ
--vault-key | --ext-vault-key
  # Vault keyまたは、外部ディレクトリVault key
--export | --import # インポートかエクスポートかを指定
--archive-location <location> # 出力先アーカイブファイル
--archive-key  # アーカイブkey
```

Java Computeから資格情報へのアクセス

- V12.0.9.0からJava Computeで資格情報にアクセスが可能
  - MbCredentialメソッドを使用してルックアップが可能
  - ルックアップのパターンごとのサンプルコードは以下を参照
    - https://www.ibm.com/docs/en/app-connect/12.0?topic=java-accessing-credentials-from-user-code

（参考）ISEでの検証結果：
サンプルコードではmyCred = MbCredential.getCredential(“ODBC”, “myDSN”);というように
MbCredential.getCredentialで指定する資格情報名を大文字表記（”ODBC”)しているが、
小文字表記にする必要あり(“odbc”)
大文字表記だと“Credential cannot be found”のエラーになる

  - server.conf.yamlで取得可能な資格情報を制御可能
    - CredentialsのuserRetrievableCredentialTypesで指定
      - ✓ 参照可能な資格情報タイプをカンマ区切りで複数指定可能
      - ✓ ‘ALL’を指定すると全資格情報へのアクセスを許可
      - ✓ ‘’を指定すると全資格情報へのアクセスを禁止

server.conf.yamlから抜粋

```
Credentials:
  #userRetrievableCredentialTypes: 'userdefined’
            # Sets the comma separated list of credential types that user code is allowed to reference.
                Default is 'userdefined'.
            # The list can be empty to disallow credential lookups from user code.
            # Set the value to 'ALL' to allow access to credentials of any type from user code.
```

ServerCredentialsの構成

ServerCredentials

- setdbparms、mqsicredentialsと同等の構成をserver.conf.yamlで構成可能
  - server.conf.yamlでの記述

```
ServerCredentials:                  # ServerCredentials下に認証情報のタイプ毎に記述する
  http:                             # 認証情報のタイプを指定
    SecProfID1:                     # 認証情報名を指定(同一のタイプ内でユニークである必要あり)
      password: "password"
      username: "joe"
    SecProfID2:
     password: "drowssap"
      username: "alice"
  rest:
    remoteApiID1:
      username: "RestUser"
      password: "RestPwd"
    remoteApiID2:
      apiKey: "RestApiKey"
```

  - 使用可能なタイプ名や、各タイプで指定可能な認証情報はマニュアルを参照
    - https://www.ibm.com/docs/en/app-connect/12.0?topic=cis-configuring-security-credentials-independent-integration-server-in-serverconfyaml-file

ServerCredentials

- コンテナ環境での構成では特に役立つ
  - Kubernetes/OpenShift環境では、ConfigMapやSecretとしてserver.conf.yamlを外部リソースとして定義することで、同一のコンテナイメージから、環境毎の構成をしたコンテナを起動する事が可能
  - 構成例
    - 準備したserver.conf.yamlを元にOpenShiftのSercretリソースとして作成する

```
$ oc create secret generic aceserver1-conf-yaml --from-file ./server.conf.yaml
secret/aceserver1-conf-yaml created
```

ServerCredentials

- 作成したSecretをVolumeマウントし、コンテナ内で扱えるようにする
  - ✓ CP4IのCertified Container Imageや、DockerHubのDeveloper版コンテナイメージは、/home/aceuser/initial-config/serverconf ディレクトリに配置されたyamlファイルをserver.conf.yamlとして読み込んだ上で独立統合サーバーが起動する構成となっている
- リソース作成例(DeploymentConfig)

```
kind: DeploymentConfig
apiVersion: apps.openshift.io/v1
spec:
(- snip -)
  template:
    spec:
      containers:
        - name: aceserver1
          image: ibm/ibmace:latest
          volumeMounts:
            - name: serverconf
              mountPath: "/home/aceuser/initial-config/serverconf"
      volumes:
        - name: serverconf
          secret:
            secretName: aceserver1-conf-yaml
            items:
            - key: serverconf
              path: server.conf.yaml
```

REST APIとWeb UI

REST API

- 資格情報の作成・参照をREST APIから行うことが可能
  - REST API経由で作成された資格情報はVaultに保存される
  - 一部の資格情報では、REST API経由で更新・削除が可能
    - 資格情報を変更するために、統合ノードの停止が必要な資格情報は対象外

- URI
  - 資格情報のタイプ毎のURIの確認

    GET `http://localhost:4414/apiv2/servers/{server}/credentials`

    - 全ての資格情報を取得できるものでは無いので注意
      - ✓ V11.0.0.10現在、一括で取得できるAPIは提供されていない
  - 資格情報毎の定義の確認

    GET `http://localhost:4414/apiv2/servers/{server}/credentials/<cred_type>-credentials`

    - <cred_type>には資格情報のタイプ名が入る

REST API

- 資格情報の定義の作成

POST `http://localhost:4414/apiv2/servers/{server}/credentials/<cred_type>-credentials`

- <cred_type>: 資格情報のタイプ

Body

- Credential Type: jdbcの場合
  - ✓ 資格情報によって必要な属性は異なるので詳細はAPI Docsを確認
    - API Docs: http://localhost:4414/apidocs/ (ホスト名、ポートはそれぞれの環境のIPまたはドメイン名)

```
{
   "name": "jdbc-credential",
   "type": "jdbc-credential",
   "properties": {
     "authType": "basic",
     "name": "jdbc-api",
     "password": "jdbcapi",
     "type": "Credential",
     "username": "jdbcapi"
   }
 }
```

REST API

- 個別の資格情報の詳細の確認

GET `http://localhost:4414/apiv2/servers/{server}/credentials/<cred_type>-credentials/<credential>`

- <cred_type>: 資格情報のタイプ
- <credential>: 参照する資格情報名、またはID

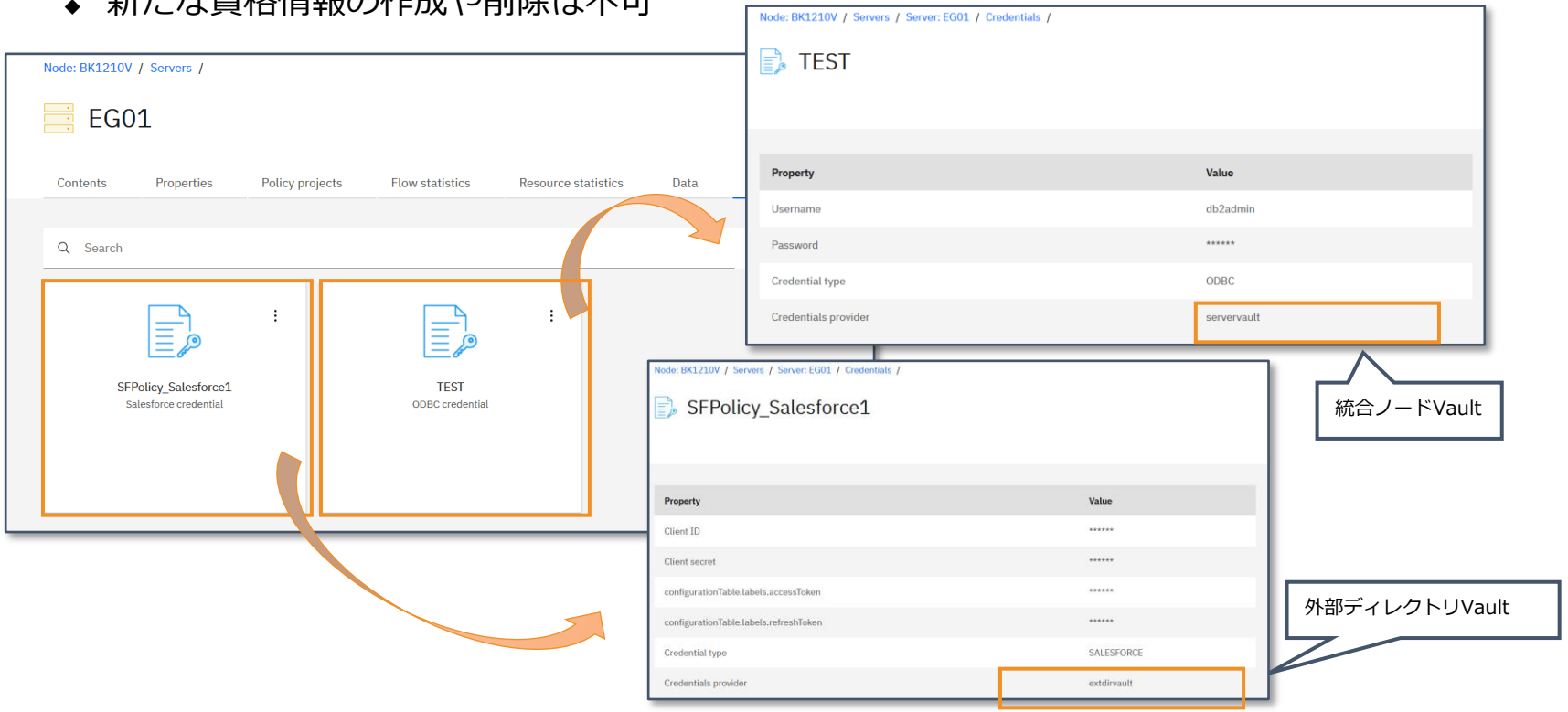
- レスポンス例

```
code: 200 OK
header:
connection: keep-alive
Content-Length: 391
Content-Type: application/json; charset=utf-8
Vary: Accept-Encoding
Date: Tue, 12 Dec 2023 07:36:22 GMT
Connection: keep-alive

{"hasChildren":false,"name":"jdbc-api","type":"jdbc-
credential","uri":"/apiv2/servers/EG01/credentials/jdbc-
credentials/jdbc-
api","properties":{"authType":"basic","credentialsProvider":"server
vault","name":"jdbc-
api","password":"******","readOnly":false,"type":"Credential","user
name":"jdbcapi"},"descriptiveProperties":{"className":"jdbc-
credential"},"active":{},"actions":{},"children":{},"links":[]}
```

Web UI

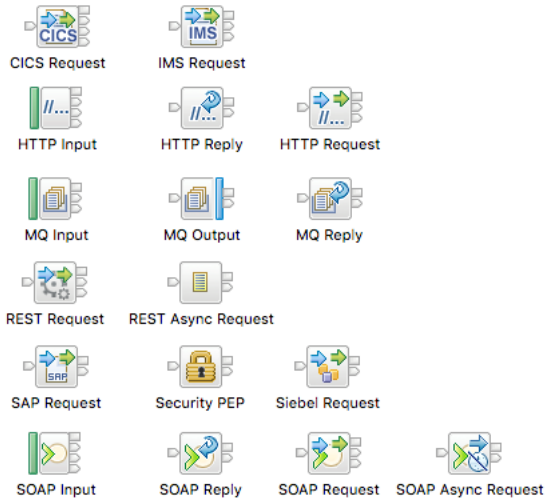
- Web UIのCredentialsタブからサーバー内で定義された資格情報を参照可能
  - 全ての資格情報の提供元(mqsisetdbparms, Vault, ServerCredentials)の情報が一覧で得られる
  - 新たな資格情報の作成や削除は不可



Vault保管の認証情報を使用したBasic認証

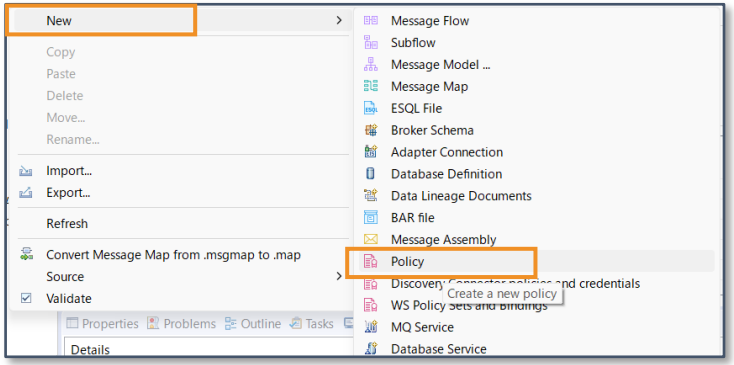
Vault保管の認証情報を使用したBasic認証 - 概要

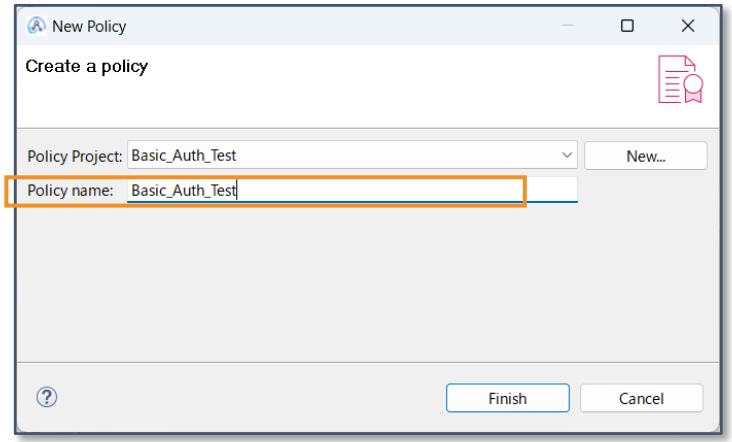
- Vaultに保管された認証情報を使用したBasic認証
  - HTTPリクエストなどにユーザー名とパスワードを要求する事が可能
    - Vaultに保管した資格情報を用いた認証が可能
      - ✓ 従来ではLDAPなどの外部セキュリティープロバイダーを使用
      - ✓ ただし、基本認証で利用するポリシーでは一組のユーザーID/パスワードしか定義できないため、システム単位での利用が現実的
    - 独立統合サーバーでのみ使用可能
  - 認証の有無の適用範囲の指定が可能
    - 独立統合サーバー全体
    - 特定のメッセージフロー全体
    - 特定のメッセージフローノード
      - ✓ 以下のノードに設定可能
        - CICSRequest
        - HTTPInput, HTTPRequest, HTTPAsyncRequest
        - IMSRequest
        - MQInput, MQOutput, MQReply
        - RESTRequest, RESTAsyncRequest
        - SAPRequest
        - SecurityPEP
        - SiebleRequest
        - SOAPInput, SOAPReply, SOAPRequest, SOAPAsyncRequest
  - V12.0.9.0以降、Basic認証の認証情報の変更をフローの再起動なしに動的に反映可能



Vault保管の認証情報を使用したBasic認証 - 構成

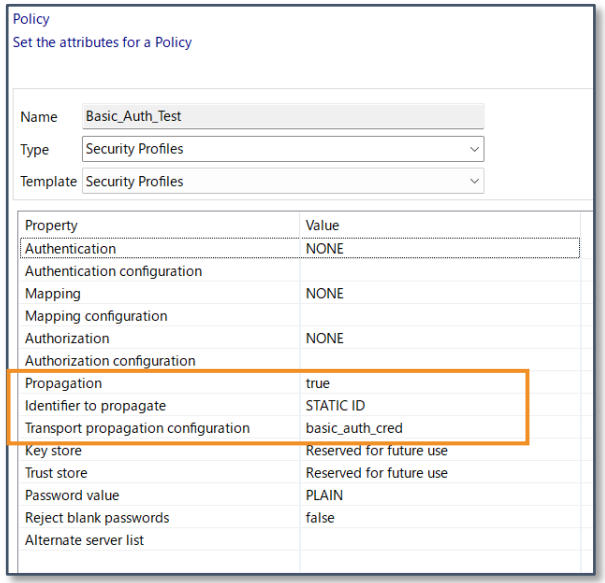
- セキュリティプロファイルポリシーの作成
  - Toolkitにて、新規作成 > ポリシーを選択
    - ポリシープロジェクトを選択または新規作成し、プロジェクトに作成するポリシー名を入力





Vault保管の認証情報を使用したBasic認証 - 構成

- ポリシーの作成
  - TypeにはSecurity Profilesを指定
  - 以下のプロパティにそれぞれ値を指定する
    - Propagation: true
    - Identifier to propagate: STATIC ID（固定）
    - Transport propagation configuration: 任意の資格情報名(ここでは basic_auth_cred として説明する)



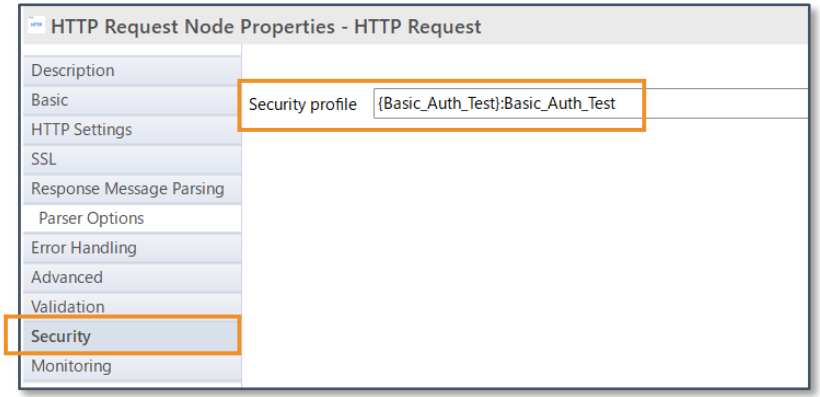
Vault保管の認証情報を使用したBasic認証 - 構成

- Vaultの作成
  - 当資料を参照
- 認証情報の追加
  - 以下の情報で資格情報をVaultに追加する(mqsicredentialsコマンド)
    - credential type: http
    - credential name: basic_auth_cred (ポリシーの Transport propagation configurationで指定した名前)
    - username: 任意
    - password: 任意

```
mqsicredentials --work-dir /home/aceuser/ace-server/ --create ¥
                      --vault-key myvaultkey ¥
                      --credential-type http ¥
                      --credential-name basic_auth_cred ¥
                      --username SecUserName ¥
                      --password SecPassword
```

Vault保管の認証情報を使用したBasic認証 - 構成

- セキュリティプロファイルを適用する
  - 特定のノードに適用する場合
    - フローのノード・プロパティのSecurityタブでSecurity profileに設定



  - 独立統合サーバー全体に適用する場合
    - server.conf.yamlに以下を追加する
      - ✓ それぞれポリシープロジェクト名、セキュリティプロファイル名に対応する

```
forceServerHTTPSecurityProfile: '{Basic_Auth_Test}:Basic_Auth_Test'
```

    - 独立統合サーバーを再起動して変更を適用する

Vault保管の認証情報を使用したBasic認証 - 構成

- フロー全体、または特定のメッセージフローノードに適用する場合
  - mqsiapplybaroverrideコマンドを使用して、メッセーフロー/メッセーフローノードのプロパティを上書きする
    - ✓ -m オプションでコマンドから直接指定するか、-p オプションでプロパティを記載したファイルを指定

```
# フロー全体に適用する場合
# Syntax:message_flow_name#SecurityProfileName={policy_project_name}:security_profile_name
MyService.msgflow#SecurityProfileName={Basic_Auth_Test}:Basic_Auth_Test
# 特定のメッセーフローノードに適用する場合
# ノード名に続いてピリオド区切りでプロパティ名(SecurityProfileName)を記述する
MyService.msgflow#HTTP Input.SecurityProfileName={Basic_Auth_Test}:Basic_Auth_Test
```

  - mqsiapplybaroverrideコマンドの実行

```
$ mqsiapplybaroverride -b Microservice1.bar ¥
                       -o Microservice1-basic.bar ¥
                       -p override.properties ¥
                       -k Microservice1
        BIP1138I: Applying overrides using runtime mqsiapplybaroverride...
        BIP1140I: Overriding property MyService.msgflow#HTTP Input.SecurityProfileName with
'{BasicAuthPolicy}:BasicAuthPolicy1' in 'Microservice1.appzip/META-INF/broker.xml' ...
        BIP1143I: Saving Bar file Microservice1-basic.bar...
        BIP8071I: Successful command completion.
```

  - 生成されたBARファイルを`mqsibar`コマンド等でデプロイする
  - 独立統合サーバーを再起動する