外部設定ファイルの認証

- 外部設定ファイルの認証 （1 ページ）
- 設定データの暗号化 （7 ページ）
- ConfD AES 暗号化文字列をエンコードするための Cisco Elastic Controller サービススクリプト （9 ページ）

外部設定ファイルの認証

Cisco ESC リリース 4.0 以前では、ESC は、デイゼロ設定、モニタリング、展開、および LCS アクションの一部として、いくつかの外部設定ファイルとスクリプトをサポートしています。ESC は、展開の一部として、認証の有無にかかわらず、リモートサーバからのこれらのファイルの取得をサポートしています。

ESC リリース 4.0 以降、ファイルロケータ属性は展開レベル、つまり展開コンテナの直下で定義されます。これにより、複数の VM グループとそのデイゼロ設定および LCS アクションが、展開内の必要な場所で同じファイルロケータを参照できるようになります。

展開データモデルの例は次のとおりです。

```
<esc_datamodel xmlns="http://www.cisco.com/esc/esc">
  <tenants>
    <tenant>
      <name>sample-tenant</name>
      <deployments>
        <deployment>
          <name>sample-deployment</name>
          <file_locators>
            <file_locator>
              <name>post_deploy_alive_script</name>
              <remote_file>
                <file_server_id>http-my-server</file_server_id>
              <remote_path>/share/qatest/vnfupgrade/lcspostdeployalive.sh</remote_path>

                <local_target>vnfupgrade/lcspostdepalive.sh</local_target>
                <persistence>FETCH_ALWAYS</persistence>
                <properties/>
              </remote_file>
            </file_locator>
            <file_locator>
              <name>asa-day0-config</name>
              <remote_file>
```

外部設定ファイルの認証

```
        <file_server_id>http-my-server</file_server_id>
        <remote_path>/share/qatest/day0/asa_config.sh</remote_path>
        <local_target>day0.1/asa_config.sh</local_target>
        <persistence>FETCH_ALWAYS</persistence>
      </remote_file>
    </file_locator>
    <file_locator>
      <name>scriptlocator</name>
      <remote_file>
        <file_server_id>dev_test_server</file_server_id>
        <remote_path>/share/users/gomoore/actionScript.sh</remote_path>
        <local_target>action/actionScript.sh</local_target>
        <persistence>FETCH_MISSING</persistence>
        <properties/>
      </remote_file>
    </file_locator>
  </file_locators>
  <policies>
    <policy>
      <name>VNFUPGRADE_POST_DEPLOY_ALIVE</name>
      <conditions>
        <condition>
          <name>LCS::POST_DEPLOY_ALIVE</name>
        </condition>
      </conditions>
      <actions>
        <action>
          <name>post_deploy_alive_action</name>
          <type>SCRIPT</type>
          <properties>
            <property>
              <name>file_locator_name</name>
              <value>post_deploy_alive_script</value>
            </property>
          </properties>
        </action>
      </actions>
    </policy>
  </policies>
<vm_group>
    <name>ASA-group</name>
    <image>ASAImage</image>
    <flavor>m1.large</flavor>
    <recovery_policy>
      <max_retries>1</max_retries>
    </recovery_policy>
    <scaling>
      <min_active>1</min_active>
      <max_active>1</max_active>
      <elastic>true</elastic>
    </scaling>
    <placement>
      <type>affinity</type>
      <enforcement>strict</enforcement>
    </placement>
    <bootup_time>120</bootup_time>
    <recovery_wait_time>60</recovery_wait_time>
    <interfaces>
      <interface>
        <nicid>0</nicid>
        <network>my-net</network>
      </interface>
    </interfaces>
    <kpi_data>
```

外部設定ファイルの認証

```
                    <kpi>
                      <event_name>VM_ALIVE</event_name>
                      <metric_value>1</metric_value>
                      <metric_cond>GT</metric_cond>
                      <metric_type>UINT32</metric_type>
                      <metric_occurrences_true>1</metric_occurrences_true>
                      <metric_occurrences_false>5</metric_occurrences_false>
                      <metric_collector>
                        <nicid>0</nicid>
                        <type>ICMPPing</type>
                        <poll_frequency>5</poll_frequency>
                        <polling_unit>seconds</polling_unit>
                        <continuous_alarm>false</continuous_alarm>
                      </metric_collector>
                    </kpi>
                  </kpi_data>
                  <rules>
              <admin_rules>
                      <rule>
                        <event_name>VM_ALIVE</event_name>
                        <action>ALWAYS log</action>
                        <action>TRUE servicebooted.sh</action>
                        <action>FALSE recover autohealing</action>
                      </rule>
                    </admin_rules>
                  </rules>
                  <config_data>
                    <configuration>
                      <dst>ASA.static.txt</dst>
                      <file_locator_name>asa-day0-config</file_locator_name>
                    </configuration>
                  </config_data>
                  <policies>
                    <policy>
                      <name>SVU1</name>
                      <conditions>

<condition><name>LCS::DEPLOY_UPDATE::PRE_VM_VOLUME_DETACH</name></condition>
                      </conditions>
                      <actions>
                        <action>
                          <name>LOG</name><type>pre_defined</type>
                        </action>
                        <action>
                          <name>pre_vol_detach</name>
                          <type>SCRIPT</type>
                          <properties>
                            <property>
                              <name>file_locator_name</name>
                              <value>scriptlocator</value>
                            </property>
                            <property>
                              <name>exit_val</name>
                              <value>0</value>
                            </property>
                          </properties>
                        </action>
                      </actions>
                    </policy>
                  </policies>
                </vm_group>
              </deployment>
            </deployments>
          </tenant>
```

外部設定ファイルの認証

```
  </tenants>
</esc_datamodel>
```

展開を実行する前に、APIを使用してリモートサーバ（ファイルサーバ）を個別に設定する必要があります。REST API と NETCONF API の両方がサポートされます。

- URL、ユーザ名を含む認証の詳細、およびパスワードを含むリモートサーバ。設定には REST または NETCONF を使用できます。

（注） ユーザ名とパスワードはオプションです。パスワードはESC内で暗号化されます。

  展開前にリモートファイルサーバを設定する必要があります。クレデンシャルは、展開中にいつでも更新できます。

- ファイルロケータが展開データモデルに追加されます。ファイルサーバへの参照と、ダウンロードするファイルへの相対パスが含まれます。

認証を使用してリモートでファイルを取得するには、以下を行う必要があります。

1. リモートサーバを追加します。
2. ファイルロケータでリモートサーバを参照します。ファイルロケータは、デイゼロおよびLCS アクションブロックの設定データの一部です。
3. 展開の一部として、ファイルロケータに基づいてデイゼロおよびライフサイクルステージ（LCS）スクリプトが取得されます。

ファイルサーバのパラメータは次のとおりです。

- id：ファイルサーバのキーと識別子として使用されます。
- base_url：サーバのアドレス。（例：http://www.cisco.com または https://192.168.10.23）
- file_server_user：サーバへの認証時に使用するユーザ名。
- file_server_password：サーバへの認証用のパスワードを含む文字列。最初に、ユーザは内部で暗号化されたクリアテキスト文字列を指定します。
- properties：将来の拡張性のための名前と値のペア。

ファイルロケータのパラメータは次のとおりです。

- name：ファイルロケータのキーおよび識別子として使用されます。
- local_file または remote_file：ファイルの場所を選択します。ローカルファイルは、ESC VM ファイルシステムにすでに存在するファイルを指定するために使用されます。remote_file は、リモートサーバから取得するファイルを指定するために使用されます。
  - file_server_id：ファイルを取得するファイルサーバオブジェクトの ID。

外部設定ファイルの認証

- remote_path：ファイルサーバオブジェクトで定義された base_url からのファイルのパス。
- local_target：ファイルを保存するためのオプションのローカル相対ディレクトリ。
- properties：必要な情報の名前と値のペア。
- persistence：ファイルストレージのオプション。値には、CACHE、FETCH_ALWAYS、および FETCH_MISSING（デフォルト）が含まれます。

- checksum：転送されるファイルの有効性を検証するために使用する、オプションの BSD スタイルのチェックサム値。

サーバ接続、ファイルの存在、チェックサムなどのファイルサーバ値の有効性が検証されます。

file_server_password フィールドとプロパティの encrypted_data フィールドの encrypted_data 値は、伝送用 AES / 128 ビットを使用して CFB モードで暗号化されます。データは、サーバへのアクセスに必要になるまで暗号化されたままになります。暗号化された値の詳細については、「設定データの暗号化」を参照してください。

ファイルサーバの例

```
<esc_datamodel xmlns="http://www.cisco.com/esc/esc">
  <file_servers>
    <file_server>
      <id>server-1</id> <!-- unique name for server -->
      <base_url>https://www.some.server.com</base_url>
      <file_server_user>user1</file_server_user>
      <file_server_password>sample_password</file_server_password>
      <!-- encrypted value -->
      <!-- properties list containing additional items in the future -->
      <properties>
        <property>
          <name>server_timeout</name>
          <value>60</value>
      <!-- timeout value in seconds, can be over-ridden in a file_locator -->
        </property>
      </properties>
    </file_server>
    <file_server>
      <id>server-2</id>
      <base_url>https://www.some.other.server.com</base_url>
          <properties>
        <property>
          <name>option1</name>
          <encrypted_value>$8$EADFAQE</encrypted_value>
        </property>
    </file_server>
  </file_servers>
</esc_datamodel>
```

デイゼロ設定の例

```
<esc_datamodel xmlns="http://www.cisco.com/esc/esc">
  <tenants><tenant>
      <name>sample-tenant</name>
      <deployments><deployment>
          <name>sample-deployment</name>
```

外部設定ファイルの認証

```
            <vm_group>
              <name>sample-vm-group</name>
              <config_data>
                <!-- exisiting configuration example - remains valid -->
                <configuration>
                  <file>file:///cisco/config.sh</file>
                  <dst>config.sh</dst>
                </configuration>
                <!-- new configuration including use of file locators -->
                <configuration>
                  <dst>something</dst>
                  <file_locators>
                    <file_locator>
                      <name>configlocator-1</name> <!-- unique name -->
                      <remote_file>
                        <file_server_id>server-1</file_server_id>
                        <remote_path>/share/users/configureScript.sh</remote_path>
                        <!-- optional user specified local silo directory -->
                        <local_target>day0/configureScript.sh</local_target>
                        <!-- persistence is an optional parameter -->
                        <persistence>FETCH_ALWAYS</persistence>
                        <!-- properties in the file_locator are only used for
                             fetching the file not for running scripts -->
                        <properties>
                          <property>
                            <!-- the property name "configuration_file" with value "true"
 indictates this is the
                                 script to be used just as using the <file> member case
of the configuration -->
                            <name>configuration_file</name>
                            <value>true</value>
                          </property>
                          <property>
                            <name>server_timeout</name>
                            <value>120</value> <!-- timeout value in seconds, overrides
the file_server property -->
                          </property>
                          </properties>
                        </remote_file>
                        <!-- checksum is an optional parameter.
                           The following algorithms are supported: SHA-1, SHA-224, SHA-256,
 SHA-384, SHA-512 -->
                        <checksum>SHA256 (configureScript.sh) =
dd526bb2c0711238ec2649c4b91598fb9a6cf1d2cb8559c337c5f3dd5ea1769e</checksum>
                      </file_locator>
                      <file_locator>
                        <name>configlocator-2</name>
                        <remote_file>
                          <file_server_id>server-2</file_server_id>
                          <remote_path>/secure/requiredData.txt</remote_path>
                          <local_target>day0/requiredData.txt</local_target>
                          <persistence>FETCH_ALWAYS</persistence>
                          <properties/>
                        </remote_file>
                      </file_locator>
                    </file_locators>
                  </configuration>
                </config_data>
              </vm_group>
            </deployment></deployments>
        </tenant></tenants>
</esc_datamodel>
```

デイゼロ設定およびLCSアクションの詳細については、「デイゼロ設定」および「再展開ポリシー」の項を参照してください。

設定データの暗号化

秘密キーと秘密情報を使用して設定データを暗号化できます。ESCでは、デイゼロ設定、デイゼロ設定変数、VIMコネクタとVIMユーザ、およびLCSアクションに秘密キーが含まれています。

ConfDは、暗号化された文字列タイプを提供します。組み込みの文字列タイプを使用すると、暗号化された値がConfDに保存されます。値の暗号化に使用されるキーは、confd.confに保存されます。

データの暗号化はオプションです。必要に応じて、encrypt_data値を使用してデータを保存できます。

次の例では、デイゼロ設定データに暗号化された値が含まれています。encrypted_dataは組み込みの文字列タイプ *tailf:aes-cfb-128-encrypted-string* を使用します。

```
choice input_method {
  case file {
    leaf file {
      type ietf-inet-types:uri;
    }
  }
  case data {
    leaf data {
      type types:escbigdata;
    }
  }
  case encrypted_data {
    leaf encrypted_data {
      type tailf:aes-cfb-128-encrypted-string;
    }
  }
}
```

Advanced Encryption Standard（AES）キーの生成

AESキーの長さは16バイトで、32文字の16進数文字列が含まれています。

暗号化を機能させるには、confd.confでAESキーを設定する必要があります。

```
/opt/cisco/esc/esc-confd/esc_production_confd.conf
```

```
<encryptedStrings>
    <AESCFB128>
        <key>0123456789abcdef0123456789abcdef</key>
        <initVector>0123456789abcdef0123456789abcdef</initVector>
    </AESCFB128>
</encryptedStrings>
```

デフォルトのAESキーはconfDで使用できます。

```
0123456789abcdef0123456789abcdef
```

confDキーはハードコードされています。escadm.pyはランダムなAESキーを生成し、confDが開始する前にデフォルトのconfD AESキーを置き換えます。

変数の暗号化

encrypted_val を使用して、デイゼロ設定内でパスワードやシャーシ ID などの変数を暗号化できます。ESC では、展開データモデル内の変数として *val* または *encrypted_val* を選択できます。

encrypted_val 内のテキストは暗号化されて、confD データベース（CDB）と PostgreSQL DB に格納されます。テキストは使用時にのみ復号化されます（データが保存されているときは復号化されません）。ESC ログでは、*encrypted_val* のテキストがマスクされます。

次の例では、ノースバウンドクライアント（Netconf または REST）で *encrypted_val* にプレーンテキストが入力されています。展開要求が ESC ConfD によって処理されると、プレーンテキストは暗号化されて ESC データベースに格納されます。

```
<config_data>
  <configuration>
    <dst>vnf_day0.cfg</dst>
    <data>file://opt/cisco/esc/esc_database/vnf_day0.cfg</file>
    <variable>
      <name>user</name>
      <val>admin</val>
    </variable>
    <variable>
      <name>password</name>
      <encrypted_val>ADMIN-PASSWORD</encrypted_val>
    </variable>
```

encrypted_val が netconf または CLI を介して ConfD 設定から取得されると、暗号化形式でプレーンテキストが表示されます。

```
<config_data>
  <configuration>
    <dst>vnf_day0.cfg</dst>
    <data>file://opt/cisco/esc/esc_database/vnf_day0.cfg</file>
    <variable>
      <name>user</name>
      <val>admin</val>
    </variable>
    <variable>
      <name>password</name>

<encrypted_val>$8$cVl6r9aR7W3wmHLYUrAOQHnjJGH0XltJjiCBTXANJFV0sJfb/NF+lEJiUA0j/JxA</encrypted_val>

    </variable>
```

（注） 単一の値が *encrypted_val* に格納されます。同じ変数値が置き換えられて、スケールグループ内にあるすべての VM のデイゼロ設定テンプレートに入力されます。

デイゼロ設定では、*encrypted_val* を使用してシャーシ ID を保護できます。シャーシ ID の値は、VNF のアップグレードを実行するノースバウンドクライアントまたはオペレータによって提供されます（VNF 展開中にスクリプトによって生成されたシャーシ ID はサポートされません）。

```
<config_data>
  <configuration>
    <dst>staros_param.cfg</dst>
    <file>file://opt/cisco/esc/images/staros_param_upf.cfg</file>
    <variable>
      <name>CHASSIS_ID</name>
      <encrypted_val>VALUE-PROVIDED-BY-NORTHBOUND-OPERATOR</encrypted_val>
    </variable>
```

デイゼロ設定の詳細については、デイゼロ設定を参照してください。

ConfD AES 暗号化文字列をエンコードするための Cisco Elastic Controller サービススクリプト

この機能は、dep.xml など、設定要求で使用できる AES 暗号化文字列をエンコードするスクリプトを提供します。次に、同じ機能を提供する 2 つのスクリプト（代替）を示します。

- `esc_nc_cli encrypt`
- `esc_confd_encrypt` ：ESC VM または ESC VM への接続が可能なリモート Linux サーバで使用するスタンドアロンスクリプトです。

次のコマンドは、プレーンテキストを AES 暗号化文字列に暗号化するのに役立ちます。

```
esc_nc_cli encrypt
```

次に例を示します。

```
admin@esc-01$ esc_nc_cli encrypt
Enter plain text (input is not echoed to terminal) >  *********
admin@127.0.0.1's password:
$8$aaCBcnVmZ+6lEV1FvhhitzQMLisLc3pxk1uUh+7DL4A=


admin@esc-01$ esc_nc_cli encrypt input.txt
admin@127.0.0.1's password:
$8$SLwFZuA0m0Rgf69fPNOeiq4ispm5H1SZIVGzzDd5R2g=
```

次のコマンドは、独立したスタンドアロンスクリプトとして実装される `esc_nc_cli` と同等です。

次に例を示します。

```
admin@esc-01$ esc_confd_encrypt
Enter plain text (input is not echoed to terminal) > ********
admin@localhost's password:
$8$QL5vFU1vt3KEs3kKIrC0+Faq8cF83WdptPO45GTIBGA=

admin@esc-01$ esc_confd_encrypt --file input.txt
admin@localhost's password:
$8$uzN7+kMgCf4RLxB5R0qMnLIbixO6EUpliUuHJRwR944=
```

次のコマンドは、ConfD CLI ssh（ポート 2024）に接続します。

次に例を示します。

```
admin@esc-01$ esc_nc_cli cli
ssh -o StrictHostKeyChecking=no -p 2024 admin@127.0.0.1
admin@127.0.0.1's password: *****

admin connected from 127.0.0.1 using ssh on esc-01
admin@esc-01>
```

リモートホストからのスクリプトの使用

両方のスクリプトを使用して、リモートESCで暗号化を実行できます。たとえば、ESC VM、ノースバウンドクライアント、または管理「ジャンプホスト」に接続できるLinuxサーバなどです。

次に例を示します。

```
abc@my-server-39:~$ esc_confd_encrypt --host 172.25.0.89 --user admin
Enter plain text (input is not echoed to terminal) >
admin@172.25.0.89's password:
$8$VUnQkT30fKqAWWCiyDPkqUjS+jDd0/sNIyGNd4bVppE=

abc@my-server-39:~$ esc_nc_cli encrypt --host 172.25.0.89 --user admin
Enter plain text (input is not echoed to terminal) >
admin@172.25.0.89's password:
$8$uRBKqpZZ9rcUIrfBam0WfCXq3tirTD+FRcafBqAArRs=

abc@my-server-39:~$ esc_nc_cli encrypt --host 172.25.0.89 --user admin --password
'REDACTED'
Enter plain text (input is not echoed to terminal) >
$8$iG9vvLAqk69wUSMVMVf5XDpwkdDi/P1V9ucJlXKn2NQ=
```

公開キー認証によるスクリプトへのパスワードレスアクセスの有効化

`esc_nc_cli`や`esc_confd_encrypt`など、ラッパーユーティリティを介して直接使用するために、ConfD（netconfおよびssh cli）に対してパスワードレスアクセス（公開キー認証）を有効にする方法は2つあります。

次に、ConfDで秘密キーペアと設定公開キー認証を作成する例を示します（推奨）。

```
admin@esc-01$ ssh-keygen -t rsa -b 2048 -C "admin" -N "" -f ~/.ssh/test_confd_rsa
Generating public/private rsa key pair.
Your identification has been saved in /home/admin/.ssh/test_confd_rsa.
Your public key has been saved in /home/admin/.ssh/test_confd_rsa.pub.
The key fingerprint is:
SHA256:u3/dpc4iY6/60fiGjGeJjMcigUKlSrxCptZWYo8JQ6o admin
The key's randomart image is:
+---[RSA 2048]----+
|                 |
| . .             |
|+ o              |
|.X o .           |
|O *.*   S        |
|Eo.=..   . o    .|
|o..  . +.+oo.. o.|
|    . o *.Xo+.o .|
|     . ooB+Booo  |
+----[SHA256]-----+
admin@esc-01$ sudo mkdir --mode=700 -p /var/confd/homes/admin/.ssh
admin@esc-01$ sudo cp ~/.ssh/test_confd_rsa.pub /var/confd/homes/admin/.ssh/authorized_keys
admin@esc-01$ sudo chown -R esc-user:esc-user /var/confd/homes/admin/.ssh
```

```
admin@esc-01$ printf "value-of-encrypted_val" | esc_nc_cli encrypt --privKeyFile
~/.ssh/test_confd_rsa
$8$VmDBKYupSGUCaILw8g2VYykVD9D16jA44sQNglFUUAv+uQtO0BmEtSC85vfuRJu0

admin@esc-01$ printf "value-of-encrypted_val" | esc_confd_encrypt --privKeyFile
~/.ssh/test_confd_rsa
$8$oFXwX1jeIHVxmBuMdPe6Vz6usaSahPVh0gZEGHm0uoAvK+twC0kUK5w7/QY0goUM

admin@esc-01$ cat .ssh/config
Host localhost 127.0.0.1
   Port 2024
   IdentityFile ~/.ssh/test_confd_rsa

admin@esc-01$ printf "value-of-encrypted_val" | esc_nc_cli encrypt
$8$GZ4+2nSo/YklKVk8RTdNR9oDJjWe89VsUiUR2FnIwtW4WPSXLivOXbmZnHR2YpfP

admin@esc-01$ printf "value-of-encrypted_val" | esc_confd_encrypt
$8$ggQaMq3QEIhS+1P8gmtr47LwdPyrCFoHHC2jzv2vKnxBFvIPNQapHurj+bcHfpEe
```

次に、組み込みの esc-nc-admin アカウントで ConfD にアクセスするために ConfD キーを有効にする例を示します（下位互換性のために提供）。

```
admin@esc-01$ sudo escadm confd keygen --user admin
Generated SSH key pair for user admin and authorized them for user esc-nc-admin

admin@esc-01$ printf "value-of-encrypted_val" | esc_nc_cli encrypt
$8$4c5m8cqK21VNyb1gCfc77p41LKxA9Ar8n6CApQwNst8yk/ilDphiDXetmHPmKuvP

admin@esc-01$ printf "value-of-encrypted_val" | esc_confd_encrypt
$8$yY8sG6leUkrnY+fBUrYVmnwPSBY9aIrUKXmpaHVGfvNWggLuSPkqZcRCjejPej+y
```