ESC 正常性のモニタリング

ESC とそのサービスの正常性を監視するには、次のいずれかを使用します。

REST API を使用した ESC の正常性のモニタリング

ESC は、ESC およびそのサービスの正常性を監視するためのサードパーティ製ソフトウェアに REST API を提供します。サードパーティ製ソフトウェアは API を使用して ESC が正常な状態であるかを定期的に照会し、ESC が稼働中であるかどうかを確認できます。クエリへの応答として、API はステータスコードとメッセージを提供します。詳細については、表 1 : スタンドアロンおよびアクティブ/スタンバイ ハイアベイラビリティにおける ESC ヘルス API のステータスコードとメッセージ （4 ページ）を参照してください。HA セットアップでは、仮想 IP（VIP）をモニタリング IP として使用する必要があります。戻り値で、ESC HA ペアの全体的な状態が示されます。詳細については、表 3 : スタンドアロン ESC と HA のヘルス API ステータスメッセージ （6 ページ）を参照してください。

ESC の正常性を監視する REST API は次のとおりです。

```
GET to https://<esc_vm_ip>:8060/esc/health
```

（注）

- ヘルス API のモニタリングは、既存の REST の基本的な HTTP 認証を使用して保護されます。ユーザは ESC REST API クレデンシャルを使用してレポートを取得できます。
- ESC ヘルス API ポート番号が 60000 から 8060 に変更されました。

次に、エラー状態のヘルス API のモニタリングの応答を示します。

JSON 応答の例：

```
<?xml version="1.0" encoding="UTF-8" ?>
```

```
<esc_health_report>
<status_code>{error status code}</status_code>
<message>{error message}</message>
</esc_health_report>
```

ローカルアクティブ/アクティブのヘルス API のモニタリングの応答は次のとおりです。

```
<?xml version="1.0" encoding="UTF-8" ?>
<esc_health_report>
    <status_code>2010</status_code>
    <message>ESC service is being provided. ESC AA cluster one or more node(s) not
healthy</message>
    <nodes>
        <node>
            <name>aa-esc-1.novalocal</name>
            <status>HEALTHY</status>
            <datacenter>dc1</datacenter>
            <services>
                <service>
                    <name>escmanager</name>
                    <status>running</status>
                    <is_expected>True</is_expected>
                </service>
                <service>
                    <name>elector</name>
                    <status>leader</status>
                    <is_expected>True</is_expected>
                </service>
                <service>
                    <name>drbd</name>
                    <status>active</status>
                    <is_expected>True</is_expected>
                </service>
                <service>
                    <name>pgsql</name>
                    <status>running</status>
                    <is_expected>True</is_expected>
                </service>
                ...
            </services>
        </node>
        <node>
            <name>aa-esc-2.novalocal</name>
            <status>HEALTHY</status>
            <datacenter>dc1</datacenter>
            <services>
                <service>
                    <name>escmanager</name>
                    <status>running</status>
                    <is_expected>True</is_expected>
                </service>
                <service>
                    <name>elector</name>
                    <status>follower</status>
                    <is_expected>True</is_expected>
                </service>
                <service>
                    <name>drbd</name>
                    <status>standby</status>
                    <is_expected>True</is_expected>
                </service>
                <service>
                    <name>pgsql</name>
                    <status>stopped</status>
                    <is_expected>True</is_expected>
```

```
                </service>

                ...
            </services>
        </node>
        <node>
            <name>aa-esc-3.novalocal</name>
            <status>NOT_HEALTHY</status>
            <datacenter>dc1</datacenter>
            <services>
                <service>
                    <name>escmanager</name>
                    <status>stopped</status>
                    <is_expected>False</is_expected>
                </service>
                <service>
                    <name>elector</name>
                    <status>follower</status>
                    <is_expected>True</is_expected>
                </service>
                <service>
                    <name>vimmanager</name>
                    <status>running</status>
                    <is_expected>True</is_expected>
                </service>
                ...
            </services>
        </node>
    </nodes>
</esc_health_report>
```

XML 応答と JSON 応答は、ヘルス API のモニタリングでもサポートされています。

API 応答が成功すると、 *stage* という追加のフィールドが導入されます。

```
<?xml version="1.0" encoding="UTF-8" ?>
<esc_health_report>
<status_code>{success status code}</status_code>
<stage>{Either INIT or READY}</stage>
<message>{success message}</message>
</esc_health_report>
```

stage フィールドには、INIT パラメータまたは READY パラメータが含まれています。

INIT：INIT パラメータは ESC が設定パラメータの設定や VIM コネクタの登録などの事前プロビジョニング要求を受け入れる初期段階のものです。

READY：ESC は、このパラメータを使用した展開、展開解除などのあらゆるプロビジョニング要求に対応できます。

ESC の正常性の状態が次のステータスコードとメッセージで示されます。2000 シリーズのステータスコードは、ESC が動作していることを意味します。5000 シリーズのステータスコードは、1 つ以上の ESC コンポーネントが稼働していないことを意味します。

表 *1:* スタンドアロンおよびアクティブ/スタンバイ ハイアベイラビリティにおける *ESC* ヘルス *API* のステータスコードとメッセージ

ステータス コード	メッセージ
2000	ESC サービスが実行されています。（ESC services are running.）
2010	ESC サービスが提供されています。（ESC services are being provided.）ESC AA クラスタの 1 つまたは複数のノードが正常ではありません。（ESC AA cluster one or more node(s) not healthy.）
2040	ESC サービスが実行されています。VIM が設定されており、ESC が VIM への接続を初期化しています。（ESC services running. VIM is configured, ESC initializing connection to VIM.）
5010	ESC サービス、ESC_MANAGER が実行されていません。（ESC service, ESC_MANAGER is not running.）
5020	ESC サービス、CONFD が実行されていません。（ESC service, CONFD is not running.）
5030	ESC サービス、MONA が実行されていません。（ESC service, MONA is not running.）
5040	ESC サービス、VIM_MANAGER が実行されていません。（ESC service, VIM_MANAGER is not running.）
[5060]	ESC サービス、ETSI が実行されていません。（ESC service, ETSI is not running.）
5070	Vim コネクタ ID [vimId_1,vimId_2,...,vimId_N] がダウンしています。（Vim Connector IDs [vimId_1,vimId_2,...,vimId_N] are down.） または 25 個のうち 6 個の VIM コネクタがダウンしています。（6 of 25 VIM Connectors are down.） （注） 6 つ以上の VIM コネクタ ID がダウンしている場合、VIM ID のリストの代わりにサマリーメッセージが出力されます。

ステータス コード	メッセージ
5080	NFVO サービスは使用できません。（The NFVO service is not available.）
5090	複数の ESC サービス（ConfD や Mona など）が実行されていません。（More than one ESC service (for example, confd and mona) are not running.）
5091	1 つ以上の ESC サービスが実行されていないため、NFVO サービスを使用できません。(One or more ESC services is not running and the NFVO service is not available.)
5092	VIM コネクタ ID [vim-1] がダウンしています。（VIM Connector ID [vim-1] is down.）NFVO サービスは使用できません。（The NFVO service is NOT available.）

表 *2:* アクティブ/アクティブ ハイアベイラビリティにおける *ESC* ヘルス *API* のステータスコードとメッセージ

ステータス コード	メッセージ
2000	ESC サービスが実行されています（アクティブ/アクティブセットアップ）。（ESC services are running (Active-Active setup).）
2010	ESC サービスが提供されています。（ESC services are provided.）ESC アクティブ/アクティブクラスタの 1 つまたは複数のノードが正常ではありません。（In ESC Active/Active cluster one or more node(s) are not healthy.）
5000	ESC サービスが提供されていません。ESC AA クラスタが正常ではありません（ESC services not being provided, ESC AA cluster not healthy）

（注） ESC HA モードでは、DRBD セットアップでのみ ESC HA を参照します。ESC HA セットアップの詳細については、『Cisco Elastic Services Controller Install Guide』を参照してください。

次の表では、スタンドアロン ESC のステータスメッセージと、成功シナリオと障害シナリオの HA について説明します。ESC のスタンドアロンおよび HA のセットアップの詳細については、『Cisco Elastic Services Controller Install Guide』を参照してください。

表 *3:* スタンドアロン *ESC* と *HA* のヘルス *API* ステータスメッセージ

	Success（成功）	**Partial Success**（一部成功）	**Failure**（失敗）
スタンドアロン Esc	応答はヘルス API のモニタリングから収集され、ステータスコードは 2000 になります。	なし	• モニタは、ヘルス API のモニタリングからの応答を取得できません。 • 応答はヘルス API のモニタリングから収集され、ステータスコードは 5000 シリーズで返されます。
HA の ESC（アクティブ/スタンバイ）	応答はヘルス API のモニタリングから収集され、ステータスコードは 2000 になります。	応答はヘルス API のモニタリングから収集され、ステータスコードは 2010 になります。これは、ESC スタンバイノードが ESC HA の ESC アクティブノードに接続できないことを示します。ただし、これはノースバウンドへの ESC サービスには影響しません。	• モニタは、2 分以上にわたってヘルス API のモニタリングの応答を取得できません。 （注） HA スイッチオーバー時の特定の期間は ESC のヘルス API のモニタリングが使用できない場合があります。モニタリングソフトウェアは、このシナリオでサービス障害を報告するように適切なしきい値を設定する必要があります。 • 応答はヘルス API のモニタリングから収集され、ステータスコードは 5000 シリーズで返されます。

	Success（成功）	Partial Success（一部成功）	Failure（失敗）
HA（アクティブ/アクティブ）のESC	応答はヘルス API のモニタリングから収集され、ステータスコードは 2000 になります。	応答はヘルス API のモニタリングから収集され、ステータスコードは 2010 になります。この状態は、ESC サービスは提供されているが、ESC AA クラスタ内の 1 つまたは複数のノードが正常ではないことを示します。ただし、これはノースバウンドへの ESC サービスには影響しません。	• ローカルアクティブ/アクティブでは、モニタが 2 分以上にわたってヘルス API のモニタリングの応答を取得できない場合です。 アクティブ/アクティブ GEO では、モニタが 7 分以上にわたってヘルス API のモニタリングの応答を取得できない場合です（Heat テンプレートの設定によって異なります）。 （注） ローカルおよび GEO スイッチオーバー時の特定の期間は ESC のヘルス API のモニタリングが使用できない場合があります。モニタリングソフトウェアは、このシナリオでサービス障害を報告するように適切なしきい値を設定する必要があります。 • GEO スイッチオーバー期間は、Heat テンプレートの設定によって異なります。デフォルトでは、スイッチオーバーはプライマリデータセンターの障害発生から 5 分後に開始されます。 応答はヘルス API のモニタリングから収集され、ステータスコードは 5000 で返されます。 （注） スイッチオーバー中は、新しいリーダーが正常になるまで、ステータスコードは一時的に 5000 で返されます。

ESC ヘルスモニタの機能拡張

ESC ヘルスモニタ API の機能が次のように拡張されています。

- ESC コンポーネントのステータスが判別されます。
- 接続と認証の詳細を簡素化するために、SNMP エージェント用の単一連絡ポイントが提供されます。

ESC モニタコンポーネントにヘルスモニタ API が実装されました。この API を使用して、ダウンした ESC コンポーネントのリストが提供されます。ヘルスモニタは、各 ESC コンポーネントのパブリックおよび内部ヘルス URL を使用して、個々のステータスを判別します。たとえば、VNFM ステータスは、ヘルスモニタが次の URL を実行して判別します。

```
https://localhost:8252/etsi/health
```

URL によって ESC コンポーネントのステータスが判別され、該当するステータスコードとステータスメッセージが SNMP トラップ通知の一部としてを返されます。

VIM 接続ステータス用の ESC ヘルスモニタ API

ESC ヘルスモニタ API が拡張され、新しい ESC ヘルスモニタ API（URL）を使用して VIM コネクタの詳細を照会できるようになりました。

```
http://<escmanager-host>:8088/escmanager/vims
```

ESC スタンドアロン型および HA 設定では、アクティブノードに対して URL が実行されます。ESC アクティブ/アクティブ設定では、すべてのノードに対して URL が実行されます

ヘルスモニタペイロードは、設定されたすべての VIM コネクタのバイナリステータスを判断するために必要な追加情報を返します。VIM コネクタのステータスは、正常またはダウンです。

ESC ヘルスモニタ API は、単一の VIM コネクタが正常かどうかを判断するために、VIM コネクタが定義されている VIM に対してクエリを実行します。クエリの結果に **CONNECTION_SUCCESSFUL** の内部ステータスが含まれる場合、その VIM コネクタは正常です。

クエリに失敗した場合、その VIM コネクタはダウンしています。

さらに、返されるステータスメッセージには、ダウンしている特定の VIM ID のカンマ区切りリストが含まれます。この例は、ESC ヘルスモニタが 2 つのダウンした VIM コネクタのペイロードを返しています。

```
{
"message": "VIM Connector IDs [vim-connector-site-1A, vim-connector-site-1C] are down.",
"status_code": "5070"
}
```

VIM コネクタの SNMP トラップ通知の詳細については、SNMP トラップ通知を使用した ESC の正常性のモニタリング （9 ページ）を参照してください。

ESC ヘルスモニタは、デフォルトでは VIM コネクタのステータスをモニタしません。ESC ヘルスモニタを有効にするには、SNMP トラップ通知 （24 ページ）の「VIM および NFVO モニタリング用の SNMP トラップの有効化」を参照してください。

NFVO 接続ステータス用の ESC ヘルスモニタ API

ESC ヘルスモニタ API は、NFVO への接続状況を判別できます。ESC は、NFVO から ESC への接続状況を照会するための API を備えています。NFVO は、標準の SOL003 定義 API クエリに応答します。URL は次のとおりです。

```
https://<vnfm-host>:8252/etsi/nfvo/health
```

NFVO が正常に認証され、SOL003 定義 API に応答する場合、NFVO は到達可能で正常です。

この例は、NFVO が設定されているが到達不能な場合に ESC ヘルスモニタが返すペイロードを示しています。

```
{
"message": "The NFVO service is NOT available.",
"status_code": "5080"
}
```

ESC ヘルスモニタは、デフォルトでは NFVO の接続ステータスをモニタしません。ESC ヘルスモニタを有効にするには、SNMP トラップ通知 （24 ページ）の「VIM および NFVO モニタリング用の SNMP トラップの有効化」を参照してください。

ETSI 展開の詳細については、*Cisco Elastic Services Controller 5.2 ETSI NFV MANO* ユーザガイドを参照してください。

SNMP トラップ通知を使用した ESC の正常性のモニタリング

また、SNMP エージェントを使用し、SNMP トラップを介してさまざまな ESC コンポーネントの正常性に関する通知を設定することもできます。このエージェントは、標準の ESC インストールの一部としてインストールされ、SNMP バージョン 2c および 3 プロトコルをサポートしています。SNMP トラップは現在、ESC で管理されている VNF ではなく、ESC 製品の状態のみをサポートしています。この項では、ESC SNMP エージェントを設定するために必要な手順について説明します。また、通知の一部としてトリガーされるイベントについても説明します。

始める前に

- **CISCO-ESC-MIB** ファイルと **CISCO-SMI MIB** ファイルがシステムで使用できることを確認します。これらのファイルは /opt/cisco/esc/snmp/mibs ディレクトリにあります。これらのファイルを SNMP マネージャマシンにダウンロードし、$HOME/.snmp/mibs ディレクトリに配置します。

- SNMP エージェントを設定します。SNMP エージェントを設定するには、次の 3 つの方法があります。これらの方法については、次の項で詳しく説明します。

SNMP エージェントの設定

SNMP トラップを受信するには、SNMP エージェントパラメータを設定します。エージェントは、この項で説明する 3 つの異なる方法を使用して設定できます。使用する最良または最適な方法は、用途によって異なります。

1. **ESC のインストール時の SNMP エージェントの有効化および設定**：

   - **BootVM によるスタンドアロンまたはアクティブ/スタンバイ HA セットアップ**

     ESC のインストール中に、次の追加パラメータを使用して SNMP エージェントを設定します。

     ```
     % bootvm.py <esc_vm_name> --image <image-name> --net <net-name> --enable-snmp-agent
      --ignore-ssl-errors
     --managers "udp:ipv4/port,udp:[ipv6]/port"
     ```

     （注） マネージャの値は、SNMP トラップが「udp:ipv4/port」または「udp:[ipv6]/port」形式で配信される場所のカンマ区切りリストです。IP とポートは実際の値に置き換える必要があります。

   - **アクティブ/アクティブ HA 設定**

     アクティブ/アクティブインストール中に SNMP エージェントを有効にできます。設定パラメータ `ignore_ssl_errors` および `managers` リストを渡して、インストール時にエージェントを設定できます。aa-params.yaml で定義するか、次のコマンドラインで渡すことができます。

     ```
     openstack stack create name-aa --template aa.yaml -e aa-params.yaml \
     --parameter nameprefix=ESC_AA \
     --parameter image_name=ESC-5_2_0_43 \
     --parameter flavor_name=m1.large \
     ...
     --parameter snmp_agent_startup: auto \
     --parameter snmp_agent_ignore_ssl_errors: true \
     --parameter snmp_agent_managers: [ "udp:ipv4/port,udp:[ipv6]/port" ]
     ```

2. **ESCADM による有効化と設定**

   - **スタンドアロンまたはアクティブ/スタンバイ HA 設定**

     ESCADM ツールを使用して、マネージャや ignoreSslErrors プロパティなどの SNMP エージェント設定パラメータを変更できます。

     ```
     sudo escadm snmp set --ignore_ssl_errors=true
     --managers="udp:ipv4/port,udp:[ipv6]/port"
     ```

   - **アクティブ/アクティブ HA 設定**

ESC ノード 1、ノード 2、ノード 4、およびノード 5 のすべてのリーダー対応ノードで、次のコマンドを実行します。

```
sudo escadm snmp set --startup=auto
```

（注） スタック更新によってノードが削除され、再作成された場合は、直前のコマンドを再実行する必要があります。

プライマリデータセンターの SNMP 対応ノード（ノード 1 および 2）でのみ、ESC サービスを再起動します。一度に 1 つのノードです。

```
sudo escadm stop
sudo escadm restart
```

リーダーノードが正常になり、SNMP エージェントが実行されたら、リーダーノードに SNMP エージェント設定を次のように追加できます。

```
sudo escadm snmp set --ignore_ssl_errors=true
--managers="udp:ipv4/port,udp:[ipv6]/port"
```

（注） `ignore-ssl-errors` パラメータは主に、ESC VM で自己署名証明書が使用される SSL エラーを防止する開発者環境用です。

マネージャの値は、SNMP トラップが「`udp:ipv4/port`」または「`udp:[ipv6]/port`」形式で配信される場所のカンマ区切りリストです。IP とポートは実際の値に置き換える必要があります。

3. 設定ファイルの更新

この設定の更新を有効にするには、SNMP エージェントがすでに有効になっている必要があります。

設定は、/opt/cisco/esc/esc_database/snmp.conf ファイルにあります。このファイルは JSON 形式です。次に例を示します。

```
{
"publicCommunities": "public",
"users": [],
"sysDescr": "admin@localhost",
"ignoreSslErrors": "yes",
"logLevel": "INFO",
"sysName": "system name",
"managers": [{
   "privPassword": "enc:95w3hE+uZ1A3vyykaPpKEw==",
   "targetEndpoint": "udp:localhost/12000",
   "privProtocol": "AES128",
   "targetCommunity": "public",
   "label": "some manager",
   "targetProtocol": "v2",
   "authProtocol": "SHA",
   "authPassword": "enc:IYt1UIW8wug3vyykaPpKEw==",
   "authentication": "authpriv",
```

```
    "username": "admin",
    "engineId": "80:00:00:00:01:02:03:04"
  }]
  }
```

構成は、ユーザー定義のコミュニティストリングのファイル */opt/cisco/esc/esc_database/snmp.conf.* にあります。このファイルは JSON 形式です。

（注） この構成は、SNMP バージョン 2c プロトコルに適用されます。

```
{
    "publicCommunities": "test",
    "users": [],
    "sysDescr": "TestSNMPAgentConfiguration SNMP Agent",
    "ignoreSslErrors": "yes",
    "logLevel": "INFO",
    "sysName": "dnd-admin-1208",
    "managers": []
}
```

以下を使用して、snmptrapd.conf 構成ファイルを構成します。

```
AuthCommunity log,execute,net test
disableAuthorization yes
format2 %V\n% Agent Address: %A \n Agent Hostname: %B (%b)\n Enterprise OID: %N \n Trap
 Sub-Type: %q \n Community/Infosec Context: %P \n Uptime: %T \n PDU Attribute/Value Pair
 Array:\n%v \n -------------- \n
```

出力：

```
[admin@dnd-admin-1208 ~]$ snmpget -v2c -c test -M +/opt/cisco/esc/snmp/mibs localhost:2001
 CISCO-ESC-MIB::escStatusMessage.0
CISCO-ESC-MIB::escStatusMessage.0 = STRING: "ESC services are running."
[admin@dnd-admin-1208 ~]$ snmpwalk -v2c -c test -M +/opt/cisco/esc/snmp/mibs
172.24.0.33:2001 CISCO-ESC-MIB::vnfm
CISCO-ESC-MIB::escStatusMessage.0 = STRING: "ESC services are running."
CISCO-ESC-MIB::escStatusCode.0 = STRING: "2000"
```

構成は、*/opt/cisco/esc/esc_database/snmp.conf.* とコンマで区切られた複数のコミュニティにあります。

```
{
    "publicCommunities": "public, foo ,bar",
    "users": [],
    "sysDescr": "TestSNMPAgentConfiguration SNMP Agent",
    "ignoreSslErrors": "yes",
    "logLevel": "INFO",
    "sysName": "dnd-admin-1208",
    "managers": []
}
```

以下を使用して、snmptrapd.conf 構成ファイルを構成します。

```
AuthCommunity log,execute,net public, foo ,bar
disableAuthorization yes
format2 %V\n% Agent Address: %A \n Agent Hostname: %B (%b)\n Enterprise OID: %N \n Trap
 Sub-Type: %q \n Community/Infosec Context: %P \n Uptime: %T \n PDU Attribute/Value Pair
 Array:\n%v \n -------------- \n
```

例：

```
[admin@dnd-admin-1208 ~]$ snmpget -v2c -c foo -M +/opt/cisco/esc/snmp/mibs localhost:2001
 CISCO-ESC-MIB::escStatusMessage.0
CISCO-ESC-MIB::escStatusMessage.0 = STRING: "ESC services are running."
[admin@dnd-admin-1208 ~]$ snmpget -v2c -c public -M +/opt/cisco/esc/snmp/mibs
localhost:2001 CISCO-ESC-MIB::escStatusMessage.0
CISCO-ESC-MIB::escStatusMessage.0 = STRING: "ESC services are running."
[admin@dnd-admin-1208 ~]$ snmpget -v2c -c bar -M +/opt/cisco/esc/snmp/mibs localhost:2001
 CISCO-ESC-MIB::escStatusMessage.0
CISCO-ESC-MIB::escStatusMessage.0 = STRING: "ESC services are running."
```

ESC SNMP MIB の定義

次の表で、ESC MIB の内容について説明します。これらの値は、snmp.conf ファイルで設定できます。

変数	Simple IOD	説明
sysName	SNMPv2-MIB::sysName.0	ESC マシンの名前を指定します。デフォルトでは、ホスト名が取得されます。
sysDescr	SNMPv2-MIB::sysDescr.0	SNMP エージェントの名前を指定します。
sysLocation	SNMPv2-MIB::sysLocation.0	ESC マシンが配置されている場所を指定します。
sysContact	SNMPv2-MIB::sysContact.0	管理者の連絡先を指定します。

次の表に、SNMP MIB のトラップエントリを示します。エンタープライズ OID は 1.3.6.1.4.1 です。

表 *4 : SNMP MIB* トラップエントリ

ノード	索引	親
cisco	9	エンタープライズ
ciscoMgmt	9	cisco
ciscoEscMIB	844	ciscoMgmt
escNotifs	0	ciscoEscMIB
escMIBObjects	1	ciscoEscMIB
vnfm	1	escMIBObjects
escStatusMessage	1	vnfm
escStatusCode	2	vnfm

ノード	索引	親
escPreviousStatusCode	3	vnfm
escPreviousStatusMessage	4	vnfm

SNMP トラップ通知の有効化

```
sudo escadm snmp start
```

ESCADM ツールを使用して、SNMP サービスを開始します。

また、ESCADM ツールを使用してステータスの取得を停止したり、SNMP エージェントの設定を変更できます。

```
sudo escadm snmp stop
sudo escadm snmp status
sudo escadm snmp restart
```

ESC での SNMP トラップの管理

この項の内容は、次のとおりです。

- ESC での SNMP 通知タイプについて
- ESC での SNMP トラップの管理（SNMP マネージャ）
- SNMP GET/WALK の例
- トラップエンドポイントの管理（SNMP マネージャ）
- HA 環境での ESC SNMP の管理
- アクティブ/アクティブ環境での ESC SNMP エージェントの管理
- ESC での自己署名証明書の管理

ESC での SNMP 通知タイプについて

次の表に、このバージョンの SNMP エージェントでサポートされているすべてのイベントを示します。これらのステータスコードとメッセージは、ESC の状態が変更された場合にのみ、登録されたマネージャに SNMP トラップを介して返されます。2000 シリーズのステータスコードは、ESC が動作していることを意味します。5000 シリーズのステータスコードは、1 つ以上の ESC コンポーネントが稼働していないことを意味します。2000 シリーズおよび 5000 シリーズのステータスコードの詳細については、「REST API を使用した ESC の正常性のモニタリング」の項を参照してください。

ステータス コード	SNMP エージェント固有のメッセージ
5100	ESC モニタ API の使用時に HTTP エラーが発生しました。
5101	ESC モニタは応答しましたが、データを理解できませんでした。
5102	エージェントは ESC モニタ API へのネットワーク接続を作成できませんでした。
5199	未処理のエラーが発生しました（詳細はメッセージに示されます）。
5210	「AA リーダーノードが変更されました」。（"**AA LEADER node change**".）ノードがリーダーになった AA 環境では、ノード上のエージェントがこの通知を送信します。ローカルリーダーの変更のみの場合です。
5200	「HAアクティブノードが変更されました」。（"HA ACTIVE node change"）ノードがアクティブノードになった A/S HA 環境では、エージェントがこの通知を送信します。
5220	「**GEO AA プライマリデータセンターが変更されました**」（"Geo AA Primary datacenter change" ）GEO A/A 環境では、GEO スイッチオーバー後にノードがリーダーになると、ノード上のエージェントがこの通知を送信します。GEO リーダーの変更のみの場合です。

ESC での SNMP トラップの管理（SNMP マネージャ）

SNMP マネージャは別のシステムに展開され、ESC SNMP エージェントに登録されます。たとえば、アシュアレンスシステムは ESC から受信した SNMP トラップの一般的なコンシューマです。

次の例では、*snmptrapd*、*snmpget*、*snmpwalk* などの基本的な UNIX SNMP ツールを使用します。

SNMPv2c の例

次のように SNMP トラップデーモンの構成ファイルを設定します。

```
authCommunity log,execute,net public
format2 %V\n% Agent Address: %A \n Agent Hostname: %B (%b)\n Enterprise OID: %N \n Trap
 Sub-Type: %q \n Community/Infosec Context: %P \n Uptime: %T \n PDU Attribute/Value Pair
 Array:\n%v \n -------------- \n
```

これにより、*snmptrapd* は「public」コミュニティストリングを使用して受信した通知を処理できます。端末セッションでデーモンを起動し、次のコマンドを実行します。

```
snmptrapd -f -C -c ./snmptrapd.conf -Le 12000
```

2 番目のセッションを開いて、トラップが受信されているかどうかを確認します。

```
snmptrap -v 2c -c public -n "" localhost:12000 0 linkUp.0
```

セッション 1 では、次のようになります。

```
Agent Address: somehost.somedomain
Agent Hostname: localhost (UDP: [127.0.0.1]:51331->[0.0.0.0]:0)
Enterprise OID: .
Trap Sub-Type: 0
Community/Infosec Context: TRAP2, SNMP v2c, community public
Uptime: 0
PDU Attribute/Value Pair Array:
DISMAN-EVENT-MIB::sysUpTimeInstance = Timeticks: (0) 0:00:00.00
SNMPv2-MIB::snmpTrapOID.0 = OID: IF-MIB::linkUp.0
--------------
```

ESC SNMP エージェントをテストし、「snmp.config」の次のマネージャエントリを使用します。SNMP エージェントによって生成されたトラップも、デーモンによってログに記録されます。シスコおよび ESC MIB が ~/.snmp/mibs に存在することを確認します。

SNMPv2 のマネージャエントリ

```
"managers": [{
    "targetEndpoint": "udp:localhost/12000",
    "targetCommunity": "public",
    "label": "Trap test v2c",
    "targetProtocol": "v2c"
}]
```

SNMPv3 の例

snmptrapd.conf ファイルを次のように更新します。

```
disableAuthorization no
authCommunity log,execute,net public

createUser -e 0x8000000001020304 admin SHA authpassword AES privpassword
authUser log admin

format2 %V\n% Agent Address: %A \n Agent Hostname: %B (%b)\n Enterprise OID: %N \n Trap
 Sub-Type: %q \n Community/Infosec Context: %P \n Uptime: %T \n PDU Attribute/Value Pair
 Array:\n%v \n -------------- \n
```

これにより、*admin* ユーザが追加されます。「-e」は、エンジン ID（5ー32 文字の 16 進数文字列）を示します。すべての SNMP v3 エージェントには、エージェントの一意の識別子として機能するエンジン ID があります。エンジン ID は、メッセージの認証および暗号化用のキーを生成するためのハッシュ関数とともに使用されます。

システムが通信するには、両側で同じ authProtocol（MD5 または SHA）と privProtocol（AES または DES）を使用する必要があります。一部のデバイスでは、これらの組み合わせのすべてはサポートされていません。トラップレシーバが同じように設定されていることを確認するに

は、どのサービスが使用可能になっているかを確認する必要があります。1 つの端末セッションでデーモンを再起動します。

```
snmptrapd -f -C -c ./snmptrapd.conf -Le 12000
```

2 番目のセッションで設定をテストし、ユーザ名、パスワード、エンジン ID などを照合します。*authPriv* セキュリティレベルでは、認証と暗号化の両方が選択されることに注意してください。

```
snmptrap -v 3 -n "" -a SHA -A authpassword -x AES -X privpassword -l authPriv -u admin
-e 0x8000000001020304 localhost:12000 0 linkUp.0
```

これにより、ウィンドウ 1 にトラップのログが記録されます。

出力例：

```
Agent Address: casper.cisco.com
 Agent Hostname: localhost (UDP: [127.0.0.1]:53434->[0.0.0.0]:0)
 Enterprise OID: .
 Trap Sub-Type: 0
 Community/Infosec Context: TRAP2, SNMP v3, user admin, context
 Uptime: 0
 PDU Attribute/Value Pair Array:
DISMAN-EVENT-MIB::sysUpTimeInstance = Timeticks: (0) 0:00:00.00
SNMPv2-MIB::snmpTrapOID.0 = OID: IF-MIB::linkUp.0
```

ESC で上記の設定を使用する際、次の例を参考にしてください。エンジン ID の数値は、トラップデーモンで使用される「0x」形式ではなく、コロンで区切られることに注意してください。

SNMPv3 のマネージャエントリ

```
"managers": [{
   "privPassword": "privpassword",
   "targetEndpoint": "udp:localhost/12000",
   "privProtocol": "AES128",
   "targetCommunity": "public",
   "label": "V3 trap test",
   "targetProtocol": "v3",
   "authProtocol": "SHA",
   "authPassword": "authpassword",
   "authentication": "authpriv",
   "username": "admin",
   "engineId": "80:00:00:00:01:02:03:04"
}],
```

v3 メッセージの ESC 出力例

```
Agent Address: casper.cisco.com
 Agent Hostname: localhost (UDP: [127.0.0.1]:52103->[0.0.0.0]:0)
 Enterprise OID: .
 Trap Sub-Type: 0
 Community/Infosec Context: TRAP2, SNMP v3, user admin, context 80:00:00:00:01:02:03:04
 Uptime: 0
 PDU Attribute/Value Pair Array:
DISMAN-EVENT-MIB::sysUpTimeInstance = Timeticks: (27252277) 3 days, 3:42:02.77
SNMPv2-MIB::snmpTrapOID.0 = OID: SNMPv2-SMI::enterprises.9.9.844.0.1
SNMPv2-MIB::sysDescr.0 = STRING: SNMP Agent
SNMPv2-SMI::enterprises.9.9.844.1.1.2.0 = STRING: "2000"
SNMPv2-SMI::enterprises.9.9.844.1.1.1.0 = STRING: "ESC services are running."
 --------------
```

トラップ出力

通常、トラップには statusCode、statusMessage、previousStatusCode、previousStatusMessage の 4 つのエントリが含まれます。

```
DISMAN-EVENT-MIB::sysUpTimeInstance = Timeticks: (3971) 0:00:39.71
SNMPv2-MIB::snmpTrapOID.0 = OID: CISCO-ESC-MIB::statusNotif
SNMPv2-MIB::sysDescr.0 = STRING: ESC SNMP Server
CISCO-ESC-MIB::escStatusCode.0 = STRING: "2000"
CISCO-ESC-MIB::escStatusMessage.0 = STRING: "ESC services are running."
CISCO-ESC-MIB::escPreviousStatusCode.0 = STRING: "5102"
CISCO-ESC-MIB::escPreviousStatusMessage.0 = STRING: "Warning: Could not connect to ESC
Monitor. See log for details."
```

ESC SNMP エージェントは、以前のステータスやステータスコードメッセージとともに SNMP トラップを送信します。これにより、クライアントは最新の SNMP トラップの応答先を判断できます。

以前のステータスコードやメッセージがない場合、これらの文字列は空になります。たとえば、SNMPエージェントは、以前のステータスコードとステータスメッセージの値を MIB 文字列として返します。

```
CISCO-ESC-MIB::escStatusCode.0 = STRING: "2000"
CISCO-ESC-MIB::escStatusMessage.0 = STRING: "ESC services are running."
CISCO-ESC-MIB::escPreviousStatusCode.0 = STRING: "5090"
CISCO-ESC-MIB::escPreviousStatusMessage.0 = STRING: "More than one ESC service (confd,
etsi) not running."
```

これにより、SNMP クライアントは、すべてのサービスが実行中であること、およびこの SNMP トラップが応答している Confd と ETSI サービスは、以前に実行されておらず、現在実行中であることを認識できます。

SNMP マネージャオプション

表 *5: SNMP* マネージャオプション

キー	プロトコル	説明
targetCommunity	v2c	トラップを送信するコミュニティ。デフォルトは *public* です。
label	v2c/v3	このマネージャの名前。
targetEndpoint	v2c/v3	トラップの送信先のアドレスとポート。例：*udp:localhost/12000*。
targetProtocol	v2c/v3	このマネージャに使用する SNMP プロトコル。v2c または v3。デフォルトは v2c です。

キー	プロトコル	説明
authPassword	v3	ユーザのパスワード。プレーンテキストのパスワードを入力すると、エージェントはそのパスワードを検出して暗号化します。
authProtocol	v3	使用する認証プロトコル。SHA または MD5。
認証	v3	認証のタイプは、AuthPriv、AuthNoPriv、または NoAuthNoPriv のいずれかです。
engineId	v3	このトラップに使用するエンジン ID（16 進数）。エンジン ID は、マネージャが使用する ID と一致させる必要があります。例：80:00:00:00:01:02:03:04
privPassword	v3	暗号化（プライバシー）パスワード。プレーンテキストのパスワードを入力します。エージェントはパスワードを検出して暗号化します。
privProtocol	v3	暗号化プロトコルは、DES、AES、AES128、AES192、または AES256 のいずれかになります。
ユーザ名	v3	認証用のユーザ名（またはセキュリティ名）。

SNMP GET/WALK の例

この項では、SNMP ツールの *snmpwalk* および *snmpget* を使用した SNMP *get* の実行方法の例を示します。

（注） この例では、ESC MIB が SNMP MIB パスに追加されていることを前提としています。

SNMP GET - コマンドラインの例

表 *6:*

SNMP の例	コマンド	出力例
SNMPv2c の例	`snmpget -v2c -c public localhost:2001 CISCO-ESC-MIB::escStatusMessage.0`	**出力例** `CISCO-ESC-MIB::escStatusMessage.0 = STRING: "ESC services are running."`

SNMPv3 NoAuthNoPriv の例	次のユーザが ESC SNMP エージェントの設定に追加されます。
	V3 SNMP のユーザエントリ
	`"users": [{ "username": "admin" }],`
	コマンド
	`snmpget -v3 -l authpriv -u admin localhost:2001 CISCO-ESC-MIB::escStatusMessage.0`
		出力例
		`CISCO-ESC-MIB::escStatusMessage.0 = STRING: "ESC services are running."`

SNMPv3 AuthNoPriv の例	次のユーザが ESC SNMP エージェントの設定に追加されます。
	V3 SNMP のユーザエントリ
	`"users": [{ "username": "admin", "authProtocol": "SHA", "authPassword": "authpassword" }],`
	コマンド
	`snmpget -v3 -l authpriv -u admin -a "SHA" -A "authpassword" localhost:2001 CISCO-ESC-MIB::escStatusMessage.0`
		出力例
		`CISCO-ESC-MIB::escStatusMessage.0 = STRING: "ESC services are running."`

SNMP の例	コマンド	出力例

SNMPv3 AuthPriv の例	次のユーザが ESC SNMP エージェントの設定に追加されます。
	`"users": [{ "username": "admin", "authProtocol": "SHA", "authPassword": "authpassword", "privProtocol": "AES128", "privPassword": "privpassword" }],`
	コマンド
	`snmpget -v3 -l authpriv -u admin -a "SHA" -A "authpassword" -x "AES128" -X "privpassword" localhost:2001 CISCO-ESC-MIB::escStatusMessage.0`
		出力例
		`CISCO-ESC-MIB::escStatusMessage.0 = STRING: "ESC services are running."`

トラップエンドポイントの管理（SNMP マネージャ）

SNMPエージェントは、構成ファイルに変更がないかを監視し、変更が行われるとリロードします。コンフィギュレーションファイルに対してマネージャのエンドポイントを追加または削除し、以降のトラップでは新しい設定が使用されます。

HA 環境での ESC SNMP エージェントの管理

2 つ以上の ESC ノードが HA 設定で展開されます。SNMP エージェントがこの設定をサポートします。ただし、HA 展開では次の点を考慮してください。

- SNMPを有効にするには、アクティブノードとスタンバイノードの両方を設定する必要があります。
- 1 つの ESC ノード（アクティブノード）のみが SNMP トラップを送信できます。
- スイッチオーバーが発生すると、スタンバイノードのSNMPエージェントは自動的にアクティブノードの設定を受信します。
- フェールオーバーが原因でスタンバイノードがアクティブノードになると、トラップが生成されます。

AA 環境での ESC SNMP エージェントの管理

SNMP エージェントサービスは、ローカルまたは GEO ESC アクティブ/アクティブ設定でもサポートされます。アクティブ/アクティブ展開における考慮事項は次のとおりです。

- SNMP エージェントはリーダーノードでのみトラップを実行し、送信します。

- トラップは次のシナリオで送信されます。
  - ESC ヘルス API のステータスコードの変更時。SNMP エージェントは、AA のヘルスモニタ API をポーリングします。返されたステータスコードに変更がある場合は、トラップとしてサブスクライバに送信されます。
  - ローカルスイッチオーバーを示す新しいリーダーになるノードによる、ローカルスイッチオーバーの後。
  - 新しい GEO プライマリデータセンターのリーダーになるノードによる、GEO スイッチオーバーの後。
- リーダーノードの設定に対する変更は、スイッチオーバー後に新しいリーダーによって引き継がれます。

専用の対象コミュニティによる複数のマネージャの管理

SNMP マネージャは別のシステムに展開され、ESC SNMP エージェントに登録されます。

ESC SNMP エージェントは、SNMP トラップを受信する複数のマネージャ構成の配列をサポートし、各構成には専用の対象コミュニティがあります。

次の例は、専用の対象コミュニティによる複数のマネージャのサポートを示しています。

ウィンドウ 1 を開く

SNMP エージェント構成ファイル */opt/cisco/esc/esc_database/snmp.conf* を開き、次の情報を追加します。

```
{
    "publicCommunities": "public",
    "users": [],
    "sysDescr": "TestSNMPAgentConfiguration SNMP Agent",
    "ignoreSslErrors": "yes",
    "logLevel": "INFO",
    "sysName": "dnd-admin-1208",
    "managers": [
        {
            "targetEndpoint": "udp:localhost/12006",
            "targetCommunity": "test1",
            "label": "Trap test v2c",
            "targetProtocol": "v2c"
        },
        {
            "targetEndpoint": "udp:localhost/12004",
            "targetCommunity": "test2",
            "label": "Trap test v2c",
            "targetProtocol": "v2c"
        }
    ]
}
```

ウィンドウ 2 を開く

以下を使用して、SNMP トラップデーモン構成ファイルを構成します。

```
AuthCommunity log,execute,net test1
```

```
disableAuthorization yes

format2 %V\n% Agent Address: %A \n Agent Hostname: %B (%b)\n Enterprise OID: %N \n Trap
 Sub-Type: %q \n Community/Infosec Context: %P \n Uptime: %T \n PDU Attribute/Value Pair
 Array:\n%v \n -------------- \n
```

ウィンドウ 3 を開く

以下を使用して、SNMP トラップデーモン構成ファイルを構成します。

```
AuthCommunity log,execute,net test2
disableAuthorization yes
format2 %V\n% Agent Address: %A \n Agent Hostname: %B (%b)\n Enterprise OID: %N \n Trap
 Sub-Type: %q \n Community/Infosec Context: %P \n Uptime: %T \n PDU Attribute/Value Pair
 Array:\n%v \n -------------- \n
```

escadm ツールを使用して、ESC VM の confd サービスを停止します。

udo escadm confd stop

escadm は、snmptrapd プロセスが「test1」コミュニティストリングを使用して通知を受信できるようにします。ウィンドウ 2 でデーモンを開始して、次のコマンドを実行します。

```
snmptrapd -f -C -c ./snmptrapdm.conf -Le 12006
```

前のコマンドを実行すると、ウィンドウ 2 に次のように表示されます。

```
Agent Address: 0.0.0.0
 Agent Hostname: dnd-admin-1208.novalocal (UDP: [127.0.0.1]:57472->[127.0.0.1]:12006)
 Enterprise OID: .
 Trap Sub-Type: 0
 Community/Infosec Context: TRAP2, SNMP v2c, community test1
 Uptime: 0
 PDU Attribute/Value Pair Array:
DISMAN-EVENT-MIB::sysUpTimeInstance = Timeticks: (1043610566) 120 days, 18:55:05.66
SNMPv2-MIB::snmpTrapOID.0 = OID: SNMPv2-SMI::enterprises.9.9.844.0.1
SNMPv2-MIB::sysDescr.0 = STRING: TestSNMPAgentConfiguration SNMP Agent
SNMPv2-SMI::enterprises.9.9.844.1.1.2.0 = STRING: "5020"
SNMPv2-SMI::enterprises.9.9.844.1.1.1.0 = STRING: "ESC service ESC_CONFD not running."
SNMPv2-SMI::enterprises.9.9.844.1.1.3.0 = STRING: "2000"
SNMPv2-SMI::enterprises.9.9.844.1.1.4.0 = STRING: "ESC services are running."
```

escadm は、snmptrapd プロセスが「test2」コミュニティストリングを使用して通知を受信できるようにします。ウィンドウ 3 でデーモンを開始して、次のコマンドを実行します。

```
snmptrapd -f -C -c ./snmptrapdm.conf -Le 12004
```

前のコマンドを実行すると、ウィンドウ 3 に次のように表示されます。

```
Agent Address: 0.0.0.0
 Agent Hostname: dnd-a-1208.novalocal (UDP: [127.0.0.1]:45804->[127.0.0.1]:12004)
 Enterprise OID: .
 Trap Sub-Type: 0
 Community/Infosec Context: TRAP2, SNMP v2c, community test2
 Uptime: 0
 PDU Attribute/Value Pair Array:
DISMAN-EVENT-MIB::sysUpTimeInstance = Timeticks: (1043610566) 120 days, 18:55:05.66
SNMPv2-MIB::snmpTrapOID.0 = OID: SNMPv2-SMI::enterprises.9.9.844.0.1
SNMPv2-MIB::sysDescr.0 = STRING: TestSNMPAgentConfiguration SNMP Agent
SNMPv2-SMI::enterprises.9.9.844.1.1.2.0 = STRING: "5020"
SNMPv2-SMI::enterprises.9.9.844.1.1.1.0 = STRING: "ESC service ESC_CONFD not running."
SNMPv2-SMI::enterprises.9.9.844.1.1.3.0 = STRING: "2000"
SNMPv2-SMI::enterprises.9.9.844.1.1.4.0 = STRING: "ESC services are running."
```

SNMP トラップ通知

VIM および NFVO モニタリング用の SNMP トラップの有効化

SNMP エージェントは ESC ヘルスモニタ API を使用して、ESC コンポーネント、VIM コネクタ、および NFVO 接続のステータスを照会します。デフォルトでは、ESC ヘルスモニタでは VIM 接続と NFVO 接続はモニタされません。また、VIM 接続と NFVO 接続に関する SNMP トラップは生成されません。

VIM および NFVO 接続ステータスの変更トラップを有効にするには、ESC ヘルスモニタの構成ファイル（`/opt/cisco/esc/esc-config/esc-config.yaml`）内に次のパラメータがあることを確認します。

```
monitor:
(2)report:
(4)nfvo:
(6)enabled: true
(4)vim_connectors:
(6)enabled: true
(6)name_threshold: 5
```

上記のパラメータが構成ファイルで指定されていない場合、VIM および NFVO 接続コンポーネントのモニタリングはデフォルトで false になります。vim_conectors および name_threshold は、一般的なメッセージの前にステータスに出力される VIM コネクタ ID の数を示します。メッセージには、ダウンしている VIM コネクタの数が示されますが、「25 個の VM コネクタの内、6 つがダウンしています。」というような詳細は示されません。

ステータスメッセージについては、「VIM コネクタの SNMP トラップ通知」を参照してください。

NFVO 接続の SNMP トラップ通知

SNMP トラップが送信されるのは、NFVO の詳細が ETSI VNFM サービス内で設定されており、ESC ヘルスモニタの設定で NFVO のモニタリングが有効になっているにもかかわらず、NFVO に到達できない場合です。

ETSI VNFM サービスは、NFVO が応答する標準 SOL003 API を使用して NFVO 接続をテストします。

NFVO に到達できない場合は、次の SNMP トラップが生成されます。

```
CISCO-ESC-MIB::escStatusCode.0 = STRING: "5080"
CISCO-ESC-MIB::escStatusMessage.0 = STRING: "The NFVO service is NOT available."
```

（注）

- NFVO に到達できるが、ログイン情報が正しくない場合、ステータスは利用不能になります。
- NFVO 接続のステータスは、ESC モニタヘルス API が実行された場合にのみ報告されます。NFVO の可用性は定期的にモニタされません。

VIM コネクタの SNMP トラップ通知

SNMP トラップが送信されるのは、VIM コネクタが ESC 内で設定されており、ESC ヘルスモニタの設定で VIM のモニタリングが 有効になっているにもかかわらず、設定されている VIM コネクタのいずれにも到達できない場合です。到達不能な VIM コネクタは、**CONNECTION_SUCCESSFUL** に相当しない内部 ESC ステータスを持つコネクタです。

- 1 つの VIM コネクタが使用できない場合は、次のトラップが生成されます。

```
CISCO-ESC-MIB::escStatusCode.0 = STRING: "5070"
CISCO-ESC-MIB::escStatusMessage.0 = STRING: "VIM Connector ID [vim-id1] is down."
```

- 2 つ以上の VIM コネクタが使用できない場合は、次のトラップが生成されます。

```
CISCO-ESC-MIB::escStatusCode.0 = STRING: "5070"
CISCO-ESC-MIB::escStatusMessage.0 = STRING: "VIM Connector IDs [vim-id1, vim-id2,
vim-id3] are down."
```

（注） VIM コネクタ数のデフォルト値は 5 です。デフォルト値は、esc-config.yaml ファイルで設定できます。「VIM および NFVO モニタリング用の SNMP トラップの有効化」を参照してください。

- 使用できない VIM コネクタ数が name_threshold を超えると、次のトラップが生成されます。

```
CISCO-ESC-MIB::escStatusCode.0 = STRING: "5070"
CISCO-ESC-MIB::escStatusMessage.0 = STRING: "6 of 25 VIM Connectors are down."
```

ESC ヘルスモニタ API の詳細については、REST API を使用した ESC の正常性のモニタリング（1 ページ）を参照してください。

結合および分割 SNMP トラップモード

SNMP エージェントは、結合または分割トラップを返すように設定されています。

- **結合トラップ**：現在、SNMP エージェントでは結合トラップが生成されます。出力が複数の ESC コンポーネントやイベントを示している場合でも、ESC ヘルスモニタからの出力が考慮されて、単一の完全なトラップとして送信されます。この出力は、SNMP エージェントの最後のポーリング期間に生成されます。複数の ESC サービスがダウンした場合も単一のトラップとして送信されます。
- **分割トラップ**：ESC リリース 5.4 以降では、各 ESC サービスやコンポーネントの**稼働**または**ダウン**イベントごとに 1 つのトラップがサポートされます。それぞれの**稼働**または**ダウン**イベントには、固有のステータスメッセージとステータスコードが割り当てられます。

（注） モニタ対象の *ESC* サービスは、既存の ESC コンポーネント（MONA、confd、ETSI、ESCMANAGER、VIMMANAGER）の正常性ステータスです。VIM コネクタの有効性と NFVO 接続は、VIM マネージャコンポーネントの一部になります（VIMMANAGER の一部としてモニタされます）。

VIM コネクタの有効性と NFVO 接続のモニタリングはともにデフォルトで無効になっています。有効にすると、ESCヘルスモニタは接続ステータスをそれぞれ自動的に報告します。SNMP エージェントはこの結果に基づいて、既存の ESC サービスと一緒にトラップを送信します。

稼働または**ダウン**イベントごとの個々のトラップが出力されると（分割トラップ）、複数の ESC サービスに対してイベントが発生したことを示すステータスコードとトラップが削除されます。そのため、次の ESC ヘルスモニタ情報は、分割モードでは SNMP トラップコードとして表示されません。ESC コンポーネント情報を結合するトラップは削除されます。

設定

結合トラップモードや分割トラップモードは *trapMode* と呼ばれる新しいプロパティによって制御されます。このプロパティは、次に示すように `/opt/cisco/esc/esc_database/snmp.conf` ファイルで設定できます。

```
{
    "publicCommunities": "public",
    "users": [],
    "sysDescr": "TestSNMPAgentTraps SNMP Agent",
    "ignoreSslErrors": "yes",
    "logLevel": "INFO",
    "sysName": "test-5-4-0-51-keep",
    "trapMode": "combined",
    "managers": []
}
```

このファイルを自動生成する場合のデフォルト値は *combined* です。これは、構成ファイルで *trapMode* が指定されていない場合もデフォルト値になります。これにより、アップグレード時の下位互換性が確保されます。

SNMP ESC コンポーネントのステータスコード

稼働 イベントトラップのステータスコード（MONA がダウンしていたが、現在は稼働状態に戻っている場合）は、新規になります。単一の ESC サービスが復元中であることを示すトラップが以前に生成されていないためです。すべての ESC サービスに対して SNMP エージェントが送信するコードのリストを以下に示します。

表 *7 : SNMP ESC* コンポーネントのステータスコード

ESC コンポーネント	稼働コード	ダウンコード	稼働コードメッセージ	ダウンコードメッセージ
すべてのサービスが稼働	2000		ESC サービスが実行されています。（ESC services are running.）	
ESC_MANAGER	2010	5010	ESC サービス、ESC_MANAGER が実行されています。（ESC service ESC_MANAGER running.）	ESC サービス、ESC_MANAGER が実行されていません。（ESC service ESC_MANAGER not running.）
ESC_CONFD	2020	5020	ESC サービス、ESC_CONFD が実行されています。（ESC service ESC_CONFD running.）	ESC サービス、ESC_CONFD が実行されていません。（ESC service ESC_CONFD not running.）
MONA	2030	5030	ESC サービス、MONA が実行されています。（ESC service MONA running.）	ESC サービス、MONA が実行されていません。（ESC service MONA not running.）
VIM_MANAGER	2040	5040	ESC サービス、VIM_MANAGER が実行されています。（ESC service VIM_MANAGER running.）	ESC サービス、VIM_MANAGER が実行されていません。（ESC service VIM_MANAGER not running.）
ETSI	2060	[5060]	ESC サービス、ETSI が実行されています。（ESC service ETSI running.）	ESC サービス、ETSI が実行されていません。（ESC service ETSI not running.）
接続サービス				

ESC コンポーネント	稼働コード	ダウンコード	稼働コードメッセージ	ダウンコードメッセージ
VIM コネクタ	2070	5070	Vim コネクタ（ID [vimid_1]）が稼働しています。（Vim Connector ID [vimid_1] is up）	Vim コネクタ（ID [vimid_1]）がダウンしています。（Vim Connector ID [vimid_1] is down.）
NFVO	2080	5080	NFVO サービスを使用できます。（The NFVO service is available.）	NFVO サービスは使用できません。（The NFVO service is NOT available.）

高可用性

ESC が高可用性ペアで動作している場合は、上記のステータスコードとメッセージが引き続き適用されますが、追加で適用できるステータスコードが 1 つあります。

表 *8*:

ESC コンポーネント	コード	メッセージ
すべてのサービスが稼働 - ESC HA ノード	2010	ESC サービスが実行されています。（ESC services are running.）ESC 高可用性ノードに到達できません。（ESC High-Availability node not reachable.）

この状況が発生すると、2010 の SNMP トラップが上記の詳細とともに送信されます。高可用性に相当する 5010 はありません。状況が解決されると、*2000 - ESC* サービスが実行されています。（*ESC services are running.*）のメッセージが送信されます。2010 ステータスコードの**稼働**トラップは送信されません。

アクティブ/アクティブ

分割モードのトラップは、アクティブ/アクティブ環境（GEO A/Aを含む）の結合モードのトラップと同じです。SNMP エージェントは、A/A の高レベルステータスコードを ESC コンポーネントごとに分割しません。

SNMP エージェントの内部トラップ

SNMP エージェントトラップは、エラーの状態に対しても送信されます。SNMP エージェントトラップは通常、内部接続エラーを示します。次の SNMP エージェントトラップは、受信時および状況が解決したときに送信されます。

表 *9 : SNMP* エージェントの内部トラップ

条件	停止/稼働コード	メッセージ
ESC ヘルスモニタ - HTTPエラー	2100/5100	ESC モニタ API の使用中に HTTP エラーを受信しました（A HTTP error was received when using the ESC Monitor API）（*HTTP* エラーについてはメッセージ内に示されます）
ESC ヘルスモニタ - 不明な応答	2101/5101	ESC モニタは応答しましたが、データを認識できませんでした（The ESC Monitor replied, but the data could not be understood）（データはメッセージ内に示されます）
ESC ヘルスモニタ - ヘルスモニタがダウン	2102/5102	ESC モニタに接続できませんでした。（Could not connect to ESC Monitor.）
ESC ヘルスモニタ - 不明なエラー	2199/5199	未処理のエラーが発生しました（An unhandled error occurred）（詳細はメッセージに示されます）
HA ノードの変更	5200	HA アクティブノードが変更されました（HA ACTIVE node change）（5200 は HA 環境でのみ有効です。「稼働」トラップと同等ではありません。SNMP エージェントが分割トラップ用に設定されている場合、HA ノードが変更されると、以前の機能と同様に 1 つの 5200 トラップのみがエンド SNMP に送信されます）

これらのコードはまれな状況を示しています。メッセージが可変長であるため、SNMP トラップ内のメッセージは（ESC コンポーネントメッセージとは異なり）変更されませんが、コードから状況と解決策を見つけることができます。5 シリーズのコードはエラーの状況を示し、2 シリーズのコードは以前の状況が修正されたことを示します。

SNMPトラップの重複と欠落

SNMP エージェントがすべての ESC コンポーネントのステータスを常にポーリングしている場合、ESCコンポーネントのステータスは保持されません。したがって、SNMPエージェントが再起動されると、ESC コンポーネントステータスの以前のビューは失われます。これにより、次の 2 つのシナリオが発生します。

- **SNMP トラップの重複**：SNMPエージェントが再起動される前にコンポーネントがダウンした場合、SNMPエージェントは重複したSNMP トラップを送信します。重複したSNMPトラップが送信されるのはまれです。

  たとえば、ESC マネージャrがダウンし、SNMP エージェントが再起動された場合、次のトラップが生成されます。

  5010 - ダウン、ESC マネージャ

  - SNMP エージェントがダウン
  - SNMP エージェントが起動して、ESC コンポーネントステータスを取得し、ESC マネージャがダウンしていることを確認すると、重複した SNMP トラップを生成

  5010 - ダウン、ESC マネージャ

- **SNMP トラップの欠落**：SNMPエージェントは、SNMPエージェントがダウンした場合にESC コンポーネントステータスの変更に対して生成されるはずの SNMP トラップを送信しない場合があります。まれに、有効なSNMPトラップを送信されない場合があります。
- たとえば、ESCマネージャがダウンし、SNMPエージェントが再起動された場合、次のトラップが生成されます。

  5010 - ダウン、ESC マネージャ

  - SNMP エージェントがダウン
  - ESC Manager が起動し、SNMP エージェントが **2010** を送信しない
  - SNMP エージェントが起動し、ステータスを取得し、ESC が正常であると認識すると、ESCマネージャの稼働トラップを送信しなかった場合でも、単一のトラップを送信

  2000 - 稼働、すべての ESC サービス

  SNMPエージェントはこのシナリオを管理するために、再起動時に常にトラップを生成します。トラップのステータスコードが「2000 - ESESC サービス、ETSI が実行されていません。（ESC service, ETSI is not running.）」の場合、エンドクライアントで以前の未確認トラップをクリアする必要があります。

自己署名証明書の管理

ESC が展開されて SNMP エージェントが ESC のヘルス API を使用する場合は、サーバにルート信頼証明書をインストールしておくことを推奨します。環境が既知であり、信頼できるものである場合、設定パラメータ「ignoreSslErrors」を使用してこれらのエラーを無視することができます。ただし、この設定をよりセキュアなデフォルトに維持する場合は、ESC 証明書を JVM 信頼ストアにインポートすることによって、自己署名証明書をインストールできます。次の項では、これを実行する手順について説明します。

ステップ 1 localhost の代替名として esc を追加します。ファイル「/etc/hosts:」で次のように追加します（または、「esc」が最後に追加されていることを確認します）。

例：

```
127.0.0.1   localhost localhost.localdomain localhost4 localhost4.localdomain4 esc
```

ステップ 2 SNMP エージェント コンフィギュレーション ファイル「/opt/cisco/esc/esc_database/snmp.conf」では、healthUrl が ESC を指している必要があります。

```
"healthUrl": "https://esc:8060:/esc/health"
```

ステップ 3 証明書をトラストストアにインポートします。次に、$JAVA_HOME is/usr/lib/jvm/jre-1.8.0-openjdk.x86_64 を想定し、証明書をインポートする例を示します。

```
cd /opt/cisco/esc/esc-config
sudo openssl x509 -inform PEM -in server.pem -outform DER -out server.cer
sudo keytool -importcert -alias esc -keystore $JAVA_HOME/lib/security/cacerts -storepass changeit
-file server.cer
```

翻訳について
このドキュメントは、米国シスコ発行ドキュメントの参考和訳です。リンク情報につきましては、日本語版掲載時点で、英語版にアップデートがあり、リンク先のページが移動/変更されている場合がありますことをご了承ください。あくまでも参考和訳となりますので、正式な内容については米国サイトのドキュメントを参照ください。