コンフィギュレーションのインポート/エクスポート

バージョンの要件：構成のインポート/エクスポートを使用するには、脅威に対する防御 バージョン 6.5(0) 以降、および 脅威に対する防御 REST API v4 以降を実行している必要があります。

Device Manager で管理されているデバイスから構成をエクスポートし、同じデバイスに、または別の互換性のあるデバイスにインポートすることができます。たとえば、構成のインポート/エクスポートを使用して、類似する複数のデバイスにベースラインの構成を複製し、各デバイスで Device Manager を使用してデバイスごとに固有の特性を設定することができます。

コンフィギュレーションのインポート/エクスポートについて

Device Manager または CDO を使用して 脅威に対する防御 デバイスをローカルで管理する場合は、脅威に対する防御 API を使用してデバイスの構成をエクスポートできます。このメソッドは、Secure Firewall Management Center が管理するデバイスでは機能しません。

構成をエクスポートすると、zip ファイルが作成されます。作成された zip ファイルはワークステーションにダウンロードできます。構成自体は、JSON 形式のテキストファイルで属性と値のペアを使用して定義されたオブジェクトとして表されます。ファイルを編集した後、同じデバイスまたは別のデバイスに再びインポートできます。

そのため、エクスポートファイルを使用してテンプレートを作成し、ネットワーク内の他のデバイスに展開できます。

オブジェクトをインポートする際、構成ファイルでオブジェクトを定義するのではなく、import コマンドで直接オブジェクトを定義することもできます。ただし、オブジェクトを直接定義するのは、少数の変更をインポートする場合に限定してください。

ここでは、構成のインポート/エクスポートについて詳しく説明します。

エクスポートファイルに含まれるもの

エクスポートを実行する場合は、どの構成をエクスポートファイルに含めるかを指定します。完全なエクスポートには、エクスポート zip ファイル内のすべてのものが含まれます。何をエクスポートするかに基づいて、エクスポート zip ファイルには次のものを含める場合があります。

- 設定された各オブジェクトを定義する属性と値のペア。Device Manager で「オブジェクト」と呼ばれるものだけに限らず、設定可能な項目はすべて、オブジェクトとしてモデル化されます。
- リモートアクセス VPN を設定した場合は、AnyConnect パッケージおよびその他の参照ファイル（クライアントプロファイル XML ファイル、DAP XML ファイル、Hostscan パッケージなど）。
- カスタムファイルポリシーを設定した場合は、すべての参照済みクリーンリストまたはカスタム検出リスト。

インポート/エクスポートとバックアップ/復元の比較

構成のインポート/エクスポートは、バックアップ/復元と同じではありません。

- バックアップ/復元は、ディザスタリカバリを目的としています。デバイスにバックアップを復元できるのは、デバイスが同じモデルであり、バックアップの取得元デバイスと同じソフトウェアバージョンを実行している場合のみです。これは主として、「最後に良好だった」構成を同じデバイスに回復すること、または構成を交換用デバイスに復元することを目的としています。
- インポート/エクスポートは、構成の全部または一部を保持することを目的としています。エクスポートファイルを使用して、デバイスのイメージを再作成した後で、構成をデバイスに復元することができます。または、エクスポートファイルをテンプレートとして使用し、その内容を編集してから別のデバイスにインポートすることもできます。インポート/エクスポートを使用すると、新しいデバイスを特定のベースラインの構成にまで迅速に設定できるため、デバイスをネットワークに迅速に導入できます。制限の範囲内で、別のデバイスモデル（たとえば、Firepower 2120 から 2130）にファイルをインポートすることもできます。インポートファイルに、すべてのデバイスモデルでサポートされているオブジェクトのみが含まれている場合、インポートに関する制限はほとんどありません。1 つ制約として、デバイスではエクスポートファイルに使用したものと同じ API バージョンを使用する必要があります。

インポート/エクスポートの戦略

次に、インポート/エクスポートを使用する方法をいくつか示します。

- **新しいデバイス用のテンプレートを作成します。**モデルデバイスを必要な基準設定にしてから、完全な構成をエクスポートします。その後、その構成を新しいデバイスにインポートしてから、Device Manager または 脅威に対する防御 API を使用して必要な変更を加えることができます。また、インポートの前にテンプレートを編集して、各インターフェイスのIPアドレスなどの変更を加えることもできます。完全なエクスポートにはManagementIPオブジェクト（type = managementip）が含まれていることに注意してください。ターゲットデバイスに管理アドレスおよびゲートウェイがすでに設定されている場合は、新しいデバイス用のテンプレートを作成するときに、エクスポートファイルからこのオブジェクトを削除する必要があります。そうしないと、管理アドレッシング情報が上書きされます。
- **あるデバイスから他の同様のデバイスに構成の変更を展開します。**たとえば、デバイスAの構成を編集するときに、いくつかの新しいネットワークオブジェクトとアクセス制御ルールを作成します。次に、保留中の変更をエクスポートし、それらの変更をデバイスBにインポートできます。両方のデバイスに構成を展開すると、同じ新しいルールが実行されます。
- **システムの再イメージ化後に構成を再適用します。**デバイスを再イメージ化すると、構成が消去されます。最初に完全な構成をエクスポートしておけば、再イメージ化の完了後にインポートすることができます。
- **ターゲットの構成を適用します。**エクスポートファイルは編集ができ、手動で作成することもできるため、別のデバイスにインポートするオブジェクトを除くすべてのオブジェクトを削除することができます。たとえば、一連のネットワークオブジェクトを含む構成ファイルを作成し、それを使用して、同じネットワークオブジェクトのグループをすべての 脅威に対する防御 デバイスにインポートすることができます。

設定のインポート/エクスポートのガイドライン

- エクスポートジョブの間は、構成データベースで書き込みロックが保持されます。ジョブが完了するまでは、APIまたはDevice Manager を使用して構成を変更することはできません。ただし、エクスポートジョブ中に、Device Manager で設定を表示したり、API でGETコールを使用したりすることは可能です。
- インポートジョブの間は、構成データベースで読み取りと書き込みの両方のロックが保持されます。ジョブが完了するまでは、APIまたはDevice Manager を使用して構成を表示したり変更を加えたりすることはできません。
- インポートされた構成は、既存の構成に追加されます。デバイスの構成を消去して、インポートした構成に置き換えることはできません。インポートの前にデバイスの構成をリセットする必要がある場合は、デバイスのCLIに移動して、**configure manager delete** コマンドを発行し、その後 **configure manager local** コマンドを発行できます。管理インターフェイスの構成のみが維持されます。
- デバイスにファイルをインポートできるのは、ファイルに含まれているメタデータオブジェクト内の apiVersion 属性で定義されているものと同じ API バージョンをデバイスで実行している場合のみです。

設定のインポートおよびエクスポート

インポート/エクスポートプロセスでは、まずローカル管理対象デバイスから構成がエクスポートされます。その後、エクスポートファイルをダウンロードし、必要に応じて編集を加え、同じデバイスまたは互換性のあるデバイスにアップロードできます。以降のトピックでは、それぞれのステップについて説明します。

設定のエクスポート

設定のエクスポートジョブを作成して開始するには、POST /action/configexport メソッドを使用します。

手順

ステップ 1 エクスポートジョブ用の JSON オブジェクト本体を作成します。

このコールで使用する JSON オブジェクトの例を次に示します。

```
{
  "diskFileName": "string",
  "encryptionKey": "*********",
  "doNotEncrypt": false,
  "configExportType": "FULL_EXPORT",
  "deployedObjectsOnly": true,
  "entityIds": [
    "string"
  ],
  "jobName": "string",
  "type": "scheduleconfigexport"
}
```

その属性は次のとおりです。

- **diskFileName**：（任意）。エクスポート zip ファイルの名前。名前を指定しない場合は、システムによって名前が生成されます。名前を指定した場合でも、一意性を確保するために名前に文字が付加される場合があります。名前の最大長は 60 文字です。
- **encryptionKey**：（任意）。zip ファイルの暗号化キー。ファイルを暗号化しない場合は、このフィールドを省略して、代わりに "doNotEncrypt": true を指定します。キーを指定する場合は、キーをワークステーションにダウンロードした後に、キーを使用して zip ファイルを開く必要があります。エクスポートされた構成ファイルでは、秘密鍵、パスワード、およびその他の機密データがクリアテキストで公開されることに注意してください（他の方法ではインポートできないため）。そのため、場合によっては暗号化キーを適用して機密データを保護する必要があります。システムでは AES 256 暗号化が使用されます。
- **doNotEncrypt**：（任意）。エクスポートファイルを暗号化するか（false）または暗号化しないか（true）。デフォルトは false です。つまり、空でない encryptionKey 属性を指定する必要があります。true を指定した場合、encryptionKey 属性は無視されます。

- **configExportType**：次のいずれかの enum 値。
  - **FULL_EXPORT**：エクスポートファイルに構成全体を含めます。これがデフォルトです。
  - **PARTIAL_EXPORT**：entityIds リストで識別されるオブジェクトとその子孫オブジェクトのみを含めます。エクスポート不可能なオブジェクトは、ID を指定しても含まれません。ユーザー定義オブジェクトはすべてエクスポート可能です。
  - **PENDING_CHANGE_EXPORT**：まだ展開されていない（つまり、保留中の変更を含む）オブジェクトのみを含めます。

- **deployedObjectsOnly**：（任意）。オブジェクトが展開されている場合にのみ、それらのオブジェクトをエクスポートファイルに含めるかどうか。つまり、保留中の変更は含めません。これらのジョブには未展開のオブジェクトしか含まれないため、PENDING_CHANGE_EXPORT ジョブではこの属性は無視されます。デフォルトは false です。これは、すべての保留中の変更がエクスポートに含まれることを意味します。保留中の変更を除外するには、true を指定します。

- **entityIds**：[ブラケット] で囲まれた、一連の開始ポイントオブジェクトの ID をカンマで区切ったリスト。このリストは PARTIAL_EXPORT ジョブでは必須です。このリスト内の各項目には、UUID 値か、または "**id=***uuid-value*"、"**type**=*object-type*"、"**name**=*object-name*" などのパターンと一致する属性と値のペアのいずれかを指定できます。たとえば、**"type=networkobject"** などを指定できます。

  type は、networkobject などのリーフエンティティ、または一連のリーフタイプのエイリアスのいずれかになります。通常の type エイリアスの例としては、network（NetworkObject と NetworkObjectGroup）、port（すべての TCP/UDP/ICMP ポート、プロトコル、およびグループタイプ）、url（URL オブジェクトおよびグループ）、ikepolicy（IKE V1/V2 ポリシー）、ikeproposal（Ike V1/V2 プロポーザル）、identitysource（すべてのアイデンティティソース）、certificate（すべての証明書タイプ）、object（Device Manager の [オブジェクト（Objects）] ページにリストされるすべてのオブジェクト/グループタイプ）、interface（すべてのネットワーク インターフェイス）、s2svpn（すべてのサイト間 VPN 関連タイプ）、ravpn（すべての RA VPN 関連タイプ）、vpn（s2svpn と ravpn の両方）などがあります。

  これらのオブジェクトとそれらから送出される参照子孫はすべて、PARTIAL_EXPORT の出力ファイルに含まれます。エクスポート不可能なオブジェクトはすべて、ID を指定した場合でも、出力から除外されます。適切なリソースタイプに対して GET メソッドを使用し、ターゲットオブジェクトの UUID、タイプ、または名前を取得します。

  たとえば、すべてのネットワークオブジェクトと、myaccessrule という名前のアクセスルール、および、UUID で識別される 2 つのオブジェクトをエクスポートする場合、次のように指定できます。

```
"entityIds": [
   "type=networkobject",
   "id=bab3e3cd-8c70-11e9-930a-1f12ee87d473",
   "name=myaccessrule",
```

```
    "acc2e3cd-8c70-11e9-930a-1f12ee87b286"
    ]",
```

- **jobName**：（任意）。エクスポートジョブの名前。ジョブの名前を指定すると、ジョブのステータスを取得するときにジョブを見つけやすくなります。
- **type**：ジョブのタイプ。常に **scheduleconfigexport** です。

例：

次の例では、export-config-1 への完全なエクスポートを実行し、他のすべての属性に対してデフォルトを受け入れます。

```
{
  "diskFileName": "export-config-1",
  "doNotEncrypt": true
  "configExportType": "FULL_EXPORT",
  "type": "scheduleconfigexport"
}
```

ステップ 2 オブジェクトをポストします。

たとえば、curl コマンドは次のようになります。

```
curl -X POST --header 'Content-Type: application/json' --header 'Accept: application/json'
 -d '{ \
   "configExportType": "FULL_EXPORT", \
   "type": "scheduleconfigexport" \
 }' 'https://10.89.5.38/api/fdm/[最新（latest）]/action/configexport'
```

ステップ 3 応答を確認します。

取得する応答コードは 200 である必要があります。最低限の JSON オブジェクトをポストした場合の正常な応答本文は次のようになります。暗号化キーを指定した場合、暗号化キーは応答でマスクされます。

```
{
  "version": null,
  "scheduleType": "IMMEDIATE",
  "user": "admin",
  "forceOperation": false,
  "jobHistoryUuid": "c7a8ba61-629a-11e9-8b8d-0fcc3c9d6d0b",
  "ipAddress": "10.24.5.177",
  "diskFileName": "export-config-1",
  "encryptionKey": null,
  "doNotEncrypt": true
  "configExportType": "FULL_EXPORT",
  "deployedObjectsOnly": false,
  "entityIds": null,
  "jobName": "Config Export",
  "id": "c79be920-629a-11e9-8b8d-85231be77de0",
  "type": "scheduleconfigexport",
  "links": {
    "self": "https://10.89.5.38/api/fdm/[最新（latest）]
/action/configexport/c79be920-629a-11e9-8b8d-85231be77de0"
```

```
  }
}
```

エクスポートジョブのステータスの確認

エクスポートジョブの完了には多少時間がかかります。構成が大きいほど、ジョブに必要な時間が長くなります。ジョブのステータスをチェックし、ファイルをダウンロードする前に正常に完了していることを確認します。

ステータスを取得するには、GET /jobs/configexportstatus を使用する方法が最も簡単です。たとえば、curl コマンドは次のようになります。

```
curl -X GET --header 'Accept: application/json'
'https://10.89.5.38/api/fdm/[最新（latest）]/jobs/configexportstatus'
```

正常に完了したジョブは次のようなステータスを返します。

```
{
  "version": "hdy62yf5xp3vf",
  "jobName": "Config Export",
  "jobDescription": null,
  "user": "admin",
  "startDateTime": "2019-04-19 13:14:54Z",
  "endDateTime": "2019-04-19 13:14:56Z",
  "status": "SUCCESS",
  "statusMessage": "The configuration was exported successfully",
  "scheduleUuid": "1ef502ad-62a5-11e9-8b8d-074ebc750708",
  "diskFileName": "export-config-1.zip",
  "messages": [],
  "configExportType": "FULL_EXPORT",
  "deployedObjectsOnly": false,
  "entityIds": null,
  "id": "1f0aad8e-62a5-11e9-8b8d-bb1ebb4d1300",
  "type": "configexportjobstatus",
  "links": {
    "self": "https://10.89.5.38/api/fdm/[最新（latest）]
/jobs/configexportstatus/1f0aad8e-62a5-11e9-8b8d-bb1ebb4d1300"
  }
}
```

または、GET /jobs/configexportstatus/{objId} メソッドを使用して特定のジョブのステータスを取得することもできます。応答オブジェクトの **id** フィールドからオブジェクト ID を取得します。

エクスポートファイルのダウンロード

エクスポートジョブが完了すると、エクスポートファイルがシステムディスクに書き込まれ、構成ファイルと呼ばれます。このエクスポートファイルは、GET /action/downloadconfigfile/{objId} メソッドを使用してワークステーションにダウンロードできます。使用可能なファイルのリストを取得するには、GET /action/configfiles メソッドを使用します。

（注） GET /action/downloadconfigfile/{objId} では、通常はオブジェクト ID としてファイル名を指定します。または、ファイルに関連付けられている ConfigExportStatus オブジェクトの ID を指定することもできます。

手順

ステップ 1 ディスク上の構成ファイルのリストを取得します。

構成ファイルのリストには、エクスポートファイルと、インポート用にアップロードしたファイルが含まれます。

curl コマンドは次のようになります。

```
curl -X GET --header 'Accept: application/json'
'https://10.89.5.38/api/fdm/[最新（latest）]/action/configfiles'
```

応答には項目のリストが表示され、これらはそれぞれが構成ファイルです。たとえば、次のリストは 2 つのファイルを示しています。すべてのファイルの **id** が default であることに注意してください。ID を無視し、代わりに **diskFileName** を使用します。

```
{
  "items": [
    {
      "diskFileName": "export-config-2.zip",
      "dateModified": "2019-04-19 13:32:28Z",
      "sizeBytes": 10182,
      "id": "default",
      "type": "configimportexportfileinfo",
      "links": {
        "self": "https://10.89.5.38/api/fdm/[最新（latest）]/action/configfiles/default"
      }
    },
    {
      "diskFileName": "export-config-1.zip",
      "dateModified": "2019-04-19 13:14:56Z",
      "sizeBytes": 10083,
      "id": "default",
      "type": "configimportexportfileinfo",
      "links": {
        "self": "https://10.89.5.38/api/fdm/[最新（latest）]/action/configfiles/default"
      }
    }
  ],
```

ステップ 2 diskFileName をオブジェクト ID として使用し、ファイルをダウンロードします。

curl コマンドは次のようになります。

```
curl -X GET --header 'Accept: application/octet-stream'
'https://10.89.5.38/api/fdm/[最新（latest）]/action/downloadconfigfile/export-config-2.zip'
```

ファイルは、デフォルトのダウンロードフォルダにダウンロードされます。APIエクスプローラからGETメソッドを発行していて、ダウンロード場所を指定するよう求めるようにブラウザが設定されている場合は、ファイルを保存するよう求められます。

正常にダウンロードされると、戻りコードが200となり、応答本文はなくなります。

エクスポートした構成ファイルの編集

構成ファイルをダウンロードした後、ファイルを解凍して、オブジェクトが含まれているテキストファイルを開くことができます。ワードパッドはメモ帳よりも読みやすい形式で内容を表示します。インストールしている他のテキストエディタを使用することもできます。独自の構成ファイルを最初から作成することもできますが、ファイル構造を理解するために構成をエクスポートする必要があります。

次のトピックでは、テキストファイルの要件について説明します。

構成ファイルの最低要件

構成ファイルには、次の最低限の要素が含まれている必要があります。

- ファイル内のオブジェクトは[ブラケット]で囲みます。ファイル全体がオブジェクトの配列になっていて、標準のJSON表記が使用されています。
- 各オブジェクトは{波カッコ}で囲みます。
- 構成ファイル内のオブジェクトを区切るには、カンマを使用します。つまり、最後のオブジェクトを除いて、オブジェクトの閉じ波カッコの後にはカンマを付ける必要があります。
- ファイル内の最初のオブジェクトはメタデータオブジェクトである必要があります。正しいオブジェクト属性を取得するには、目的のモデルのデバイスから構成をエクスポートするのが最も簡単です。たとえば、次に示すのはSecure Firewall Threat Defense Virtualデバイスから取得したメタデータオブジェクトです。デバイスをインポートする前に、構成タイプとエクスポートタイプを編集できます。必要に応じて、generatedOn属性を削除できます。

```
{"hardwareModel":"Cisco Firepower Threat Defense for VMWare",
 "type":"metadata",
 "configType":"FULL_CONFIG",
 "apiVersion":"[最新（latest）]",
 "generatedOn":"Fri Apr 19 13:32:28 UTC 2019",
 "exportType":"FULL_EXPORT",
 "softwareVersion":"6.5.0-10480"}
```

- メタデータオブジェクトでは、正しい構成タイプ（configType）値を指定する必要があります。
  - FULL_CONFIG：このテキストファイルにはデバイスの完全な構成が含まれています。

- DELTA_CONFIG：このテキストファイルには部分的な構成が含まれています。数個のオブジェクトしか含まれていない場合もあります。

- exportType は、FULL_EXPORT、PARTIAL_EXPORT、PENDING_CHANGE_EXPORT のいずれかです。
- 完全な構成をインポートする場合、メタデータオブジェクトでは、hardwareModel、softwareVersion、apiVersion の各属性を指定する必要があります。
- オブジェクトは1行または複数行に記述できますが、オブジェクト内の属性間には空の行やコメント行を入れないでください。ファイル内ではコメントは使用できません。
- オブジェクトは依存関係の順序でエクスポートされますが（別のオブジェクトによって参照されるオブジェクトが先に定義されます）、インポートの構成ファイルでその順序を維持する必要はありません。オブジェクト間の関係は、オブジェクトの名前と ID が依存する側のオブジェクト間で正しく解決されれば、インポート時に自動的に解決されます。

アイデンティティ ラッパー オブジェクトの基本構造

構成ファイルでは、アイデンティティ ラッパー オブジェクトを使用して、エクスポートまたはインポートが可能な ConfigEntity または ManagementEntity オブジェクトを定義します。次に、アイデンティティ ラッパー オブジェクトの基本構造を示します。

```
{
    "type" : "identitywrapper",
    "data" : {},
    "parentName" : "container-name",
    "oldName" : "old-object-name",
    "action" : "EDIT", //Enum values: CREATE, EDIT or DELETE
    "index" : integer,
  }
```

オブジェクトには次の属性が含まれています。

- **type**：これは常に **identitywrapper** です。
- **data**これは、ネットワークオブジェクトやアクセス制御ルールなど、構成から取得したオブジェクトを定義する属性と値のペアの集合です。このコレクションに必要な属性は、特定のオブジェクトタイプに対応するモデルと、実行しようとするアクションによって異なります。属性と値のペアは {波カッコ} で囲みます。データ配列内の属性はカンマで区切ります。
- **parentName**：（必要な場合）。限られた数のオブジェクトが ContainedObjects です。ContainedObject は、自身を含むオブジェクトとの関係を持ちます。たとえば、アクセスルール、手動 NAT ルール、サブインターフェイスなどがあります。これらの項目について、parentName は、その項目が含まれているオブジェクト（親）の名前を指定します。この属性は、含まれている側のオブジェクトに対して指定します。含まれている側のオブジェクト以外のオブジェクトには指定しないでください。また、場合によってはこれらのオブジェクトの指数を指定する必要もあります。

親が AccessPolicy などの単一のオブジェクトである（つまり、複数のオブジェクトを作成できない）場合は、実際にこの属性を省略でき、参照はシステムによって解決されます。

- **oldName**：（必要な場合）。既存のオブジェクトの名前を変更する場合は、この属性で古い名前を指定し、data 属性の **name** 属性に新しい名前を指定できます。この属性を使用するには、action を EDIT にする必要があります。
- **action**：定義されたオブジェクトに関連して実行するアクション。完全なエクスポートでは、アクションは常に **CREATE** になります。保留中の変更または部分的なエクスポートの場合は、他のアクションが **EDIT** または **DELETE** になる場合があります。

  インポート用のファイルを編集する場合は、目的のアクションを指定します。CREATE を指定したもののオブジェクトがすでに存在する場合は、アクションが EDIT に変更されます。オブジェクトが存在しない場合は、EDIT が CREATE に変更されます。DELETE アクションは変更されません。オブジェクト参照は、オブジェクトのタイプと名前、またはオブジェクトのタイプと古い名前、あるいはオブジェクトのタイプと親の名前に基づいて解決されます。

  - CREATE：これは新しいオブジェクトです。オブジェクトを POST するときに必要なデータ属性を指定する必要があります。**name** が、指定したタイプの既存のオブジェクトと一致する場合、アクションは自動的に EDIT に変更されます。

    新しいオブジェクトを作成して他のオブジェクトからそのオブジェクトを参照する場合（ネットワークオブジェクトを定義してからアクセスルールで使用する場合など）、オブジェクトの **name** が正確に参照されている必要があります。
  - EDIT：オブジェクトを更新しようとしています。オブジェクトを PUT するときに必要なデータ属性（バージョンと ID を除きます）を指定する必要があります。名前とオブジェクトタイプは更新するオブジェクトの決定に使用され、バージョン属性は常に無視されます。
  - DELETE：オブジェクトを削除しようとしています。**type** および **name** 属性をオブジェクトデータで指定する必要があります。

- **index**：（任意。整数）。アクセス制御ルールや手動 NAT ルールなどの順序付きリストの一部となっているオブジェクトの場合における、ポリシー内のオブジェクトの位置。新しいルールを作成する場合、指数値を指定しなければ、ルールは最後のルールとしてポリシーの末尾に追加されます。ルールを編集する場合は、ルールの既存の位置が保持されます。

例：別のデバイスにインポートするためのネットワークオブジェクトの編集

各オブジェクトの構成は次のようになっています。この例は、syslog サーバーの IP アドレスを定義するネットワーク ホスト オブジェクトです。

```
{"type":"identitywrapper",
 "action":"CREATE",
 "data":{
    "version":"lfxdbtbyg4ex6",
    "name":"syslog-host",
```

```
    "subType":"HOST",
    "value":"10.100.10.10",
    "isSystemDefined":false,
    "dnsResolution":"IPV4_AND_IPV6",
    "id":"2cd0ea03-62a7-11e9-8b8d-dbf377c781d8",
    "type":"networkobject"}}
```

デバイスからこのオブジェクトをエクスポートし、そのオブジェクトを別のデバイスにインポートすることを考えます。ただし、新しいデバイスでは別のアドレス（192.168.5.15）にあるsyslogサーバーを使用する必要があるとします。新しいオブジェクトを作成しようとしているので、data属性から**version**属性と**id**属性を削除します。また、**isSystemDefined**（デフォルトはfalse）と**dnsResolution**（FQDNオブジェクトの場合にのみ該当）を削除することもできます。結果として得られる新しいオブジェクトは次のようになります。

```
{"type":"identitywrapper",
 "action":"CREATE",
 "data":{
    "name":"syslog-host",
    "subType":"HOST",
    "value":"192.168.5.15",
    "type":"networkobject"}}
```

ファイルの上部で、メタデータオブジェクトを保持（または追加）する必要があります。また、行の戻り値を追加して、ファイルの内容をスキャンして確認しやすくすることもできます。以上により、完成した構成ファイルは次のようになります。

```
[
{"hardwareModel":"Cisco Firepower Threat Defense for VMWare",
 "type":"metadata",
 "configType":"DELTA_CONFIG",
 "apiVersion":"[最新（latest）]",
 "exportType":"PARTIAL_EXPORT",
 "softwareVersion":"6.5.0-10465"}
,
{"type":"identitywrapper",
 "action":"CREATE",
 "data":{
    "name":"syslog-host",
    "subType":"HOST",
    "value":"192.168.5.15",
    "type":"networkobject"}}
]
```

インポートファイルのアップロード

構成ファイルをデバイスにインポートするには、最初にそのファイルをデバイスにアップロードしておく必要があります。zipファイルまたはテキストファイルのいずれかをアップロードできます。zipファイルを使用する場合は、AnyConnectパッケージとクライアントプロファイルを含めることができます。

ファイルをアップロードするにはPOST /action/uploadconfigfileリソースを使用します。名前の最大長は60文字です。

- API エクスプローラからこのメソッドを使用する場合は、**fileToUpload** 属性の横にある **[ファイルの選択（Choose File）]** ボタンをクリックし、ワークステーションドライブからファイルを選択します。
- 独自のプログラムからメソッドを使用する場合、要求ペイロードには、file-name フィールドを含む単一の file-item を含める必要があります。ファイル名拡張子は .txt または .zip のいずれかにする必要があり、実際のファイル内容の形式とファイル拡張子に合わせる必要があります。

curl コマンドは次のようになります。

```
curl -F 'fileToUpload=@./import-1.txt'
'https://10.89.5.38/api/fdm/[最新（latest）]/action/uploadconfigfile'
```

正常に転送されると、戻りコード 200 が返され、応答本文は次のようになります。この例には、インポートジョブに必要となる 脅威に対する防御 システムのファイル名（**diskfilename**）を示しています。

```
{
  "diskFileName": "import-1.txt",
  "dateModified": "2019-04-22 10:18:12Z",
  "sizeBytes": 267,
  "id": "default",
  "type": "configimportexportfileinfo",
  "links": {
    "self": "https://10.89.5.38/api/fdm/[最新（latest）]/action/uploadconfigfile/default"

  }
}
```

設定のインポートとジョブステータスの確認

構成ファイルを 脅威に対する防御 システムにアップロードした後、構成ファイルで定義されているオブジェクトを 脅威に対する防御 の構成にインポートできます。POST /action/configimport メソッドを使用します。

オブジェクトをインポートする際、構成ファイルでオブジェクトを定義するのではなく、import コマンドで直接オブジェクトを定義することもできます。ただし、オブジェクトを直接定義するのは、1 ～ 2 個のネットワークオブジェクトの場合など、少数の変更をインポートする場合に限定してください。

手順

ステップ 1 インポートジョブ用の JSON オブジェクト本体を作成します。

このコールで使用する JSON オブジェクトの例を次に示します。

```
{
  "diskFileName": "string",
  "encryptionKey": "*********",
```

```
  "preserveConfigFile": true,
  "autoDeploy": true,
  "allowPendingChange": true,
  "excludeEntities": [
     "string"
     ]",
  "inputEntities": [
    {
      "action": "CREATE",
      "oldName": "string",
      "parentId": "string",
      "parentName": "string",
      "index": 0,
      "data": {
        "version": "string",
        "id": "string",
        "type": "identity"
      },
      "id": "string",
      "type": "IdEntityWrapper"
    }
  ],
  "jobName": "string",
  "type": "scheduleconfigimport"
}
```

その属性は次のとおりです。

- **diskFileName**：インポートする構成の zip または txt ファイルの名前。
- **encryptionKey**：zip ファイルの暗号化に使用するキー（存在する場合）。構成ファイルが暗号化されていない場合は、キーを指定しないでください。
- **preserveConfigFile**：（任意）。インポートジョブが正常に完了した後に、脅威に対する防御ディスクにインポートされた構成ファイルのコピーを保持するかどうか。ファイルを保持する場合は true を指定し、ファイルを 脅威に対する防御 ディスクから削除する場合は false を指定します。デフォルトは False です。
- **autoDeploy**：（任意）。インポートが成功した場合に展開ジョブを自動的に開始するかどうか。インポートされたオブジェクトは保留中の変更であり、変更を正常に展開するまではアクティブになりません。展開ジョブを自動的に開始するには、true を指定します。false を指定した場合は、手動で変更を展開する必要があります。デフォルトは False です。
- **allowPendingChange**：（任意）。既存の保留中の変更がある場合にインポートジョブの開始を許可するかどうかを指定します。この属性を true に設定し、autoDeploy を true に設定した場合、自動展開ジョブには、既存とインポート済みの両方の変更がすべて含まれます。この属性を false に設定すると、保留中の変更がある場合にインポートジョブは実行されません。デフォルトは False です。
- **excludeEntities**：（任意）。インポートしないオブジェクトを識別する文字列に一致するオブジェクトのリスト。インポートする必要がない項目がインポートファイルに含まれている場合（つまり、アップロードしたファイルからそれらのオブジェクトを削除しない場合）にのみ、この属性を指定する必要があります。このリスト内の各項目は、"**id**=*uuid-value*"、"**type**=*object-type*"、"**name**=*object-name*" などのパターンを持ちます。これらのパターンのいずれかに一致する入力オブジェクトが、インポートから除外されます。

type は、networkobject などのリーフエンティティ、または一連のリーフタイプのエイリアスのいずれかになります。通常の type エイリアスの例としては、network（NetworkObject と NetworkObjectGroup）、port（すべての TCP/UDP/ICMP ポート、プロトコル、およびグループタイプ）、url（URL オブジェクトおよびグループ）、ikepolicy（IKE V1/V2 ポリシー）、ikeproposal（Ike V1/V2 プロポーザル）、identitysource（すべてのアイデンティティソース）、certificate（すべての証明書タイプ）、object（Device Manager の [オブジェクト（Objects）] ページにリストされるすべてのオブジェクト/グループタイプ）、interface（すべてのネットワーク インターフェイス）、s2svpn（すべてのサイト間 VPN 関連タイプ）、ravpn（すべての RA VPN 関連タイプ）、vpn（s2svpn と ravpn の両方）などがあります。

たとえば、すべてのネットワークオブジェクト、および、名前 myobj と UUID で識別される他の 2 つのオブジェクトをインポートから除外するには、次のように指定します。

```
"excludeEntities": [
   "type=networkobject",
   "name=myobj",
   "id=acc2e3cd-8c70-11e9-930a-1f12ee87b286"
   ]",
```

- **inputEntities**：インポートするオブジェクトの数が少ない場合は、それらを構成ファイルで定義するのではなく inputEntities オブジェクトリストで定義できます。この属性を使用する場合は、diskFileName 属性を含めることはできません。または、この属性を null に設定する必要があります。
- **jobName**：（任意）。エクスポートジョブの名前。ジョブの名前を指定すると、ジョブのステータスを取得するときにジョブを見つけやすくなります。
- **type**：ジョブのタイプ。常に **scheduleconfigimport** です。

例：

次に、import-1.txt という名前の構成ファイルをインポートする例を示します。

```
{
  "diskFileName": "import-2.txt",
  "preserveConfigFile": true,
  "autoDeploy": true,
  "allowPendingChange": true,
  "type": "scheduleconfigimport"
}
```

ステップ 2 オブジェクトをポストします。

たとえば、curl コマンドは次のようになります。

```
curl -X POST --header 'Content-Type: application/json' --header 'Accept: application/json'
 -d '{ \
   "diskFileName": "import-2.txt", \
   "preserveConfigFile": true, \
   "autoDeploy": true, \
   "allowPendingChange": true, \
   "type": "scheduleconfigimport" \
 }' 'https://10.89.5.38/api/fdm/[最新（latest）]/action/configimport'
```

ステップ 3 応答を確認します。

取得する応答コードは 200 である必要があります。最低限の JSON オブジェクトをポストした場合の正常な応答本文は次のようになります。暗号化キーを指定した場合、暗号化キーは応答でマスクされます。

```
{
  "version": null,
  "scheduleType": "IMMEDIATE",
  "user": "admin",
  "forceOperation": false,
  "jobHistoryUuid": "7e360139-6725-11e9-abb5-078014531401",
  "ipAddress": "10.24.127.37",
  "diskFileName": "import-2.txt",
  "encryptionKey": null,
  "preserveConfigFile": true,
  "autoDeploy": true,
  "allowPendingChange": true,
  "jobName": "Config Import",
  "id": "7e2b52d8-6725-11e9-abb5-5dec35337506",
  "type": "scheduleconfigimport",
  "links": {
    "self": "https://10.89.5.38/api/fdm/[最新（latest）]
/action/configimport/7e2b52d8-6725-11e9-abb5-5dec35337506"
  }
}
```

ステップ 4 インポートジョブのステータスを確認するには、GET /jobs/configimportstatus を使用します。

または、GET /jobs/configimportstatus/{objId} を使用して、1 つのインポートジョブのステータスを取得することもできます。objId については、応答本文の jobHistoryUuid 値を POST /action/configimport コールで使用します。

curl コマンドは次のようになります。

```
curl -X GET --header 'Accept: application/json'
'https://10.89.5.38/api/fdm/[最新（latest）]/jobs/configimportstatus'
```

インポートに成功すると、応答本文は次のようになります。インポートに失敗した場合は、必要に応じてファイルを編集して形式や内容のエラーを修正してからやり直す必要があります。

```
{
      "version": "pcgccfnk4hmiz",
      "jobName": "Config Import",
      "jobDescription": null,
      "user": "admin",
      "startDateTime": "2019-04-25 06:43:54Z",
      "endDateTime": "2019-04-25 06:44:01Z",
      "status": "SUCCESS",
      "statusMessage": "The configuration was imported successfully",
      "scheduleUuid": "7e2b52d8-6725-11e9-abb5-5dec35337506",
      "diskFileName": "import-2.txt",
      "messages": [],
      "preserveConfigFile": true,
      "autoDeploy": true,
      "allowPendingChange": true,
      "id": "7e360139-6725-11e9-abb5-078014531401",
      "type": "configimportjobstatus",
      "links": {
```

```
        "self": "https://10.89.5.38/api/fdm/[最新（latest）]
/jobs/configimportstatus/7e360139-6725-11e9-abb5-078014531401"
  }
}
```

次のタスク

autoDeploy を false に設定した場合は、インポートした変更を組み込むために展開ジョブを実行する必要があります。POST /operational/deploy メソッドを使用します。true に設定する場合は、すでに構成が正常に展開されている必要があります。Device Manager または API（GET /operational/auditevents）で、監査ログを確認できます。展開ジョブは「Post Configuration Import Deployment」という名前です。

（注） 機能によっては特定のライセンスが必要です。たとえば、デバイスにはすべてのリモートアクセス VPN 機能用のライセンスが必要です。ただし、インポートプロセスではライセンスは検証されません。そのため、ライセンスにより制御される機能のオブジェクトを、必要なライセンスを持たないデバイスにインポートすると、展開ジョブは失敗します。この問題が発生した場合は、必要なライセンスをデバイスに割り当てるか、オブジェクトを削除してください。

不要なインポート/エクスポートファイルの削除

エクスポートジョブによって作成された構成ファイルや、構成のインポート用にアップロードした構成ファイルが不要になった場合は、ファイルを削除できます。

DELETE /action/configfiles/{objId} を使用し、objId 値としてファイル名を指定します。

たとえば、export-config-2.zip という名前のファイルを削除する curl コマンドは次のようになります。

```
curl -X DELETE --header 'Accept: application/json'
'https://10.89.5.38/api/fdm/[最新（latest）]/action/configfiles/export-config-2.zip'
```

成功すると、戻りコードが 204 となり、応答本文はありません。

GET /action/configfiles を使用して、ファイルが削除されたことを確認できます。

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。