一般的なタスクの実行

この章の内容は、次のとおりです。

一般的なタスクの実行

（注） API を使用した多くの一般的なタスクの詳細な手順については、『*Cisco APIC Getting Started Guide*』で確認できます。

ここでは、テナントを作成し、アプリケーションを展開するのに必要な一般的なタスクの簡単な例を示します。 これらは一般的な手順です。

- ログイン：セッションを開き設定タスクを実行します。
- テナントの作成：アプリケーションをホストするテナント会社のファブリック内でプライベート ドメインを作成します。
- ユーザの作成：テナントのアプリケーションを導入し維持するテナントの管理者のアカウントを作成します。
- アプリケーションの導入：エンドポイント グループとコントラクトをテナント ネットワーク内で設定し、アプリケーションをホストするのに必要なサーバを接続します。

（注） この例では、JSON および XML 構造が読みやすいように改行、スペース、インデントで展開されています。

ログイン

API セッションの認証と維持 に説明するように、データ構造内の有効なユーザ名とパスワードを **aaaLogin** API メソッドに送信することによって APIC REST API にログインできます。 ログイン成功後に、定期的にセッションを更新する必要があります。

次の例に、XML および JSON を使用して管理者としてログインし、設定時にセッションを更新し、ログアウトする方法を示します。

例：XML API を使用したユーザの認証

次に、ユーザ名とパスワードを含む XML データ構造を使用してユーザをログインする例を示します。

```
POST https://192.0.20.123/api/aaaLogin.xml

<aaaUser name="georgewa" pwd="paSSword1"/>
```

次に、ログイン成功時の応答例を示します。

```
<imdata>
  <aaaLogin
    token=
"mk/i+LHZMhFQ47YyMlfZOjmUQQV4PXwFAY/6Dyg5Bm7PC0SVdm
MVWeOlLw69AoNoRVziR8GQeNqPbX/Vu92PxWyPjGx7kOQpoTaUY
P4Y2NkBwRfl5fEDwUe5WnGl3VoBXF4nrc8QSG8N1UWCuIFIyt9O
OzgsLG58fTf4OPCUqQL="
    refreshTimeoutSeconds="300" lastName="Washington" firstName="George">
      <aaaUserDomain name="all" rolesR="0" rolesW="2">
        <aaaReadRoles/>
        <aaaWriteRoles>
          <role name="admin"/>
        </aaaWriteRoles>
      </aaaUserDomain>
      <aaaUserDomain name="common" rolesR="0" rolesW="1">
        <aaaReadRoles/>
        <aaaWriteRoles>
          <role name="read-only"/>
        </aaaWriteRoles>
      </aaaUserDomain>
    <DnDomainMapEntry dn="uni/tn-common" readPrivileges="0" writePrivileges="3"/>
  </aaaLogin>
</imdata>
```

応答には、ユーザに関するドメインとロール情報とともにセッションのタイムアウト時間とセッション トークンが含まれます。

次に、**aaaRefresh** API メソッドを使用してセッションをリフレッシュする例を示します。

```
GET https://192.0.20.123/api/aaaRefresh.xml
```

次に、セッションのリフレッシュが成功したときの応答例を示します。

```
<?xml version="1.0" encoding="UTF-8"?>
<imdata>
  <aaaLogin
    token=
"mk/i+LHZMhFQ47YyMlfZOua51veZ5mdbJAbHG7vjKVJ7QAuUDW
YGGkAdwSxjjcPYK/4Hav3QdZN/E7zJ/Rhc87tmpihVWpkP42VqU
D4aeELVQULz97ZpKnGfF0ki9vqBYkCYRg0vMstXO8IhxNYLtlC8
rxzgZFti+6biMQ+boRx="
    refreshTimeoutSeconds="0"
    lastName=""
    firstName=""/>
</imdata>
```

次の例では、**aaaLogout API** メソッドを使用してユーザをログアウトします。

```
POST https://192.0.20.123/api/aaaLogout.json

<aaaUser name="admin" />
```

次に、ログアウト成功時の応答例を示します。

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

例：JSON API を使用したユーザの認証

次に、ユーザ名とパスワードを含む JSON データ構造を使用してユーザをログインする例を示します。

```
POST https://192.0.20.123/api/aaaLogin.json

{
  "aaaUser":{
    "attributes":{
      "name":"georgewa",
      "pwd":"paSSword1"
    }
  }
}
```

次に、ログイン成功時の応答例を示します。

```
{
  "imdata":[{
      "aaaLogin":{
        "attributes":{
          "token":"token":
"GkZl5NLRZJl5+jqChouaZ9CYjgE58W/pMccR+LeXmdO0obG9NB
Iwo1VBo7+YC1oiJL9mS6I9qh62BkX+Xddhe0JYrTmSG4JcKZ4t3
bcP2Mxy3VBmgoJjwZ76ZOuf9V9AD6Xl83lyoR4bLBzqbSSU1R2N
IgUotCGWjZt5JX6CJF0=",
          "refreshTimeoutSeconds":"300",
          "lastName":"Washington",
          "firstName":"George"
```

```
        },
        "children":[{
            "aaaUserDomain":{
              "attributes":{
                "name":"all",
                "rolesR":"0",
                "rolesW":"2"
              },
              "children":[{
                  "aaaReadRoles":{
                    "attributes":{}
                  }
                }, {
                  "aaaWriteRoles":{
                    "attributes":{},
                    "children":[{
                        "role":{
                          "attributes":{
                            "name":"admin"
                          }
                        }
                      }
                    ]
                  }
                }
              ]
            }
          }, {
            "aaaUserDomain":{
              "attributes":{
                "name":"common",
                "rolesR":"0",
                "rolesW":"1"
              },
              "children":[{
                  "aaaReadRoles":{
                    "attributes":{}
                  }
                }, {
                  "aaaWriteRoles":{
                    "attributes":{},
                    "children":[{
                        "role":{
                          "attributes":{
                            "name":"read-only"
                          }
                        }
                      }
                    ]
                  }
                }
              ]
            }
          }, {
            "DnDomainMapEntry":{
              "attributes":{
                "dn":"uni/tn-common",
                "readPrivileges":"0",
                "writePrivileges":"3"
              }
            }
          }
        ]
      }
    }
  ]
}
```


応答には、ユーザに関するドメインとロール情報とともにセッションのタイムアウト時間とセッション トークンが含まれます。

次に、**aaaRefresh** API メソッドを使用してセッションをリフレッシュする例を示します。

```
GET https://192.0.20.123/api/aaaRefresh.json
```

次に、セッションのリフレッシュが成功したときの応答例を示します。

```
{
  "imdata":[{
      "aaaLogin":{
        "attributes":{
          "token":
"Zs8jf2rcseiiYydx9WbyRVOScWs6c0Orti7EBg1H02jt0JV9oe
icX1ULLquI+LCN/93cAf46AErXpT4kN0mrCQRcCMRAl6m0ObT/r
G4D9UTm6/NNnqzhntCWTKt9IovTFDpD8Ocx1oXn2jpmS6PYDBJt
Wp4rIAu1lsdM8BA24g6=",
          "refreshTimeoutSeconds":"0",
          "lastName":"",
          "firstName":""
        }
      }
    }
  ]
}
```

次の例では、**aaaLogout** API メソッドを使用してユーザをログアウトします。

```
POST https://192.0.20.123/api/aaaLogout.json

{
  "aaaUser":{
    "attributes":{
      "name":"georgewa"
    }
  }
}
```

次に、ログアウト成功時の応答例を示します。

```
{
  "imdata":[]
}
```

テナントの追加

テナントは仮想ファブリック内のポリシー所有者です。テナントは、プライベートエンティティまたは共有エンティティにすることができます。たとえば、安全に分割されたプライベートテナントまたは他のテナントにより共有されるコンテキストおよびブリッジ ドメインを持つテナントを作成できます。テナントの共有タイプは通常、common、default または infra という名前が付いています。

管理情報モデルでは、テナントはクラス fv:Tenant の管理対象オブジェクト（MO）で表されます。『*Cisco APIC Management Information Model Reference*』に従って、fv:Tenant クラスのオブジェクトはポリシー解決のユニバース（uni）クラスの子で、`uni/tn-[name]` の識別名（DN）形式を持っています。

次の例に、XML および JSON を使用して ExampleCorp という名前の新しいテナントを追加する方法を示します。

例：XML API を使用したテナントの追加

新しいテナントを作成するには、メッセージの本文または URI でクラスと十分な命名情報を指定する必要があります。

XML API を使用して ExampleCorp という名前の新しいテナントを作成するには、次の HTTP POST メッセージを送信します。

```
POST https://192.0.20.123/api/mo/uni.xml

<fvTenant name="ExampleCorp"/>
```

または、次の例のように URI でテナントに名前を付けることができます。

```
POST https://192.0.20.123/api/mo/uni/tn-ExampleCorp.xml

<fvTenant />
```

（`?rsp-subtree=modified` を POST URI に追加することで）応答が要求された場合は、成功した操作により次の応答本文が返されます。

```
<imdata>
    <fvTenant
        instanceId="0:0"
        childAction="deleteNonPresent"
        dn="uni/tn-ExampleCorp"
        lcOwn="local"
        name="ExampleCorp"
        replTs="never"
        rn=""
        status="created"
    />
</imdata>
```

テナントを削除するには、次の HTTP DELETE メッセージを送信します。

```
DELETE https://192.0.20.123/api/mo/uni/tn-ExampleCorp.xml
```

または、次の例のように、fv:Tenant 属性内に十分な命名情報と `status="deleted"` が含まれる HTTP POST メッセージを送信できます。

```
POST https://192.0.20.123/api/mo/uni.xml

<fvTenant name="ExampleCorp" status="deleted"/>
```

例：JSON API を使用したテナントの追加

新しいテナントを作成するには、メッセージの本文または URI でクラスと十分な命名情報を指定する必要があります。

JSON API を使用して新しいテナントを作成するには、次の HTTP POST メッセージを送信します。

```
POST https://192.0.20.123/api/mo/uni.json

{
```

```
  "fvTenant" : {
    "attributes" : {
      "name" : "ExampleCorp"
    }
  }
}
```

または、次の例のように URI でテナントに名前を付けることができます。

```
POST https://192.0.20.123/api/mo/uni/tn-ExampleCorp.json

{
  "fvTenant" : {
    "attributes" : {
    }
  }
}
```

（`?rsp-subtree=modified` を POST URI に追加することで）応答が要求された場合は、成功した操作により次の応答本文が返されます。

```
{
  "imdata" :
  [{
      "fvTenant" : {
        "attributes" : {
          "instanceId" : "0:0",
          "childAction" : "deleteNonPresent",
          "dn" : "uni/tn-ExampleCorp",
          "lcOwn" : "local",
          "name" : "ExampleCorp",
          "replTs" : "never",
          "rn" : "",
          "status" : "created"
        }
      }
    }
  ]
}
```

テナントを削除するには、次の HTTP DELETE メッセージを送信します。

```
DELETE https://192.0.20.123/api/mo/uni/tn-ExampleCorp.json
```

または、次の例のように、fv:Tenant 属性内に十分な命名情報と `"status" : "deleted"` が含まれる HTTP POST メッセージを送信できます。

```
POST https://192.0.20.123/api/mo/uni.json

{
  "fvTenant" : {
    "attributes" : {
      "name" : "ExampleCorp",
      "status" : "deleted"
    }
  }
}
```

ユーザの追加

APICユーザアカウントには、名前とパスワードに加えて、ユーザの権限レベル（ロール）とファブリック内のユーザ制御の範囲（ドメイン）に関する情報が含まれます。 たとえば、クラウドの管理者はテナントを作成して追加の管理者を割り当てる機能を含むグローバルな制御が可能な一方で、テナントの管理者は通常、テナントのドメインまたはネットワークに制限されます。 ユーザ アカウントを設定するときは、パスワードの有効期限ポリシーなどの追加の設定項目を指定したり、それらの項目を省略してデフォルトの設定を使用することができます。

管理情報モデルでは、APIC ユーザはクラス aaa:User の管理対象オブジェクト（MO）で表されます。『*Cisco APIC Management Information Model Reference*』に従って、aaa:User クラスのオブジェクトには `uni/userext/user-[name]` の識別名（DN）形式があります。 aaa:User クラスの直接プロパティ（属性）には、ユーザ ドメインまたは権限レベルのフィールドが含まれませんが、このクラスのリファレンスは aaa:UserDomain の子クラス（サブツリー）を示し、それには aaa:UserRole の子クラスがあります。 新しいユーザを追加する API コマンド構造には、次の階層でこれらの子クラスにユーザ ドメインおよび権限レベルの設定を含めることができます。

- aaa:User：ユーザ オブジェクトには、1 つ以上のユーザ ドメインの子オブジェクトを含めることができます。
  - aaa:UserDomain：ユーザ ドメイン オブジェクトには、1 つ以上のユーザ ロール オブジェクトを含めることができます。
    - aaa:UserRole：ユーザ ロール オブジェクトは、ユーザの権限レベルを指定します。

次の例に、XML および JSON を使用して新しいユーザを管理者権限を持つユーザ ドメイン ExampleCorp に追加する方法を示します。 追加の例に、コマンドラインからタスクを実行するための cURL コマンドの使用方法を示します。

例：XML API を使用したユーザの追加

新しいオブジェクトを作成するには、URI またはメッセージの本文でクラスと十分な命名情報を指定する必要があります。 この例では、URI には DN が含まれ、メッセージの本文にはオブジェクト クラスが含まれます。

XML API を使用して新しいユーザを作成および設定するには、次の HTTP POST メッセージを送信します。

```
POST https://192.0.20.123/api/mo/uni/userext/user-georgewa.xml

<aaaUser
  pwd="password1"
  firstName="George"
  lastName="Washington"
  phone="4085551212"
  email="georgewa@example.com" >
    <aaaUserDomain name="ExampleCorp" >
        <aaaUserRole name="admin" />
    </aaaUserDomain>
```

```
</aaaUser>
```

または、/api/mo.xml に POST 送信して DN 属性 `dn="uni/userext/user-georgewa"` を提供するか、または /api/mo/uni/userext.xml に POST 送信して名前属性 `name="georgewa"` を提供することができます。

このコマンド構造の子セクションでは、新しいユーザがそのユーザ ドメインの管理者ロールを持つExampleCorpユーザ ドメイン内に配置されるように設定されます。 新しいユーザを作成するときは、この例に示す情報より少ない情報または多い情報を指定できます。 追加属性は、後続の操作で設定できます。

（`?rsp-subtree=modified` を POST URI に追加することで）応答が要求された場合は、成功した操作により次の応答本文が返されます。

```
<imdata>
  <aaaUser
    instanceId="0:0"
    accountStatus="active"
    childAction="deleteNonPresent"
    clearPwdHistory="no"
    descr=""
    dn="uni/userext/user-georgewa"
    email="georgewa@example.com"
    encPwd=""
    expiration="never"
    expires="no"
    firstName="George"
    intId="none"
    lastName="Washington"
    lcOwn="local"
    name="georgewa"
    phone="4085551212"
    pwd="password1"
    pwdLifeTime="no-password-expire"
    pwdSet="no"
    replTs="never"
    rn=""
    status="created" >
      <aaaUserDomain
        instanceId="0:0"
        childAction="deleteNonPresent"
        descr="" intId="none"
        lcOwn="local"
        name="common"
        replTs="never"
        rn="userdomain-common"
        status="created" >
          <aaaUserRole
            instanceId="0:0"
            childAction="deleteNonPresent"
            descr=""
            intId="none"
            lcOwn="local"
            name="read-only"
            privType="readPriv"
            replTs="never"
            rn="role-read-only"
            status="created" />
          </aaaUserDomain>
      <aaaUserDomain
        instanceId="0:0"
        childAction="deleteNonPresent"
        descr=""
        intId="none"
        lcOwn="local"
        name="ExampleCorp"
        replTs="never"
```

```
        rn="userdomain-ExampleCorp"
        status="created" >
          <aaaUserRole
            instanceId="0:0"
            childAction="deleteNonPresent"
            descr=""
            intId="none"
            lcOwn="local"
            name="admin"
            privType="readPriv"
            replTs="never"
            rn="role-admin"
            status="created" />
      </aaaUserDomain>
  </aaaUser>
</imdata>
```

この応答は、ユーザが 2 つのユーザ ドメインに作成され追加されたことを示します。 要求通りに、ユーザにはExampleCorpユーザ ドメインでの管理者権限があります。 デフォルトでは、ユーザには共通のユーザ ドメインでの読み取り専用権限もあります。

ユーザを削除するには、次の HTTP DELETE メッセージを送信します。

```
DELETE https://192.0.20.123/api/mo/uni/userext/user-georgewa.xml
```

または、次の例のように、aaa:User属性内に十分な命名情報と `status="deleted"` が含まれるHTTP POST メッセージを送信できます。

```
POST https://192.0.20.123/api/mo.xml

<aaaUser dn="uni/userext/user-georgewa" status="deleted" />
```

例：JSON API を使用したユーザの追加

新しいオブジェクトを作成するには、URI またはメッセージの本文でクラスと十分な命名情報を指定する必要があります。 この例では、URI には DN が含まれ、メッセージの本文にはオブジェクト クラスが含まれます。

JSON API を使用して新しいユーザを作成および設定するには、次の HTTP POST メッセージを送信します。

```
POST https://192.0.20.123/api/mo/uni/userext/user-georgewa.json

{
  "aaaUser" : {
    "attributes" : {
      "pwd" : "password1",
      "firstName" : "George"
      "lastName" : "Washington",
      "phone" : "4085551212",
      "email" : "georgewa@example.com"
    },
    "children" : [{
        "aaaUserDomain" : {
          "attributes" : {
            "name" : "ExampleCorp"
          },
          "children" : [{
              "aaaUserRole" : {
                "attributes" : {
                  "name" : "admin"
```

```
                }
              }
            }
          ]
        }
      }
    ]
  }
}
```

または、/api/mo.json に POST 送信して DN 属性 `"dn":"uni/userext/user-georgewa"` を提供するか、または /api/mo/uni/userext.json に POST 送信して名前属性 `"name":"georgewa"` を提供することができます。

このコマンド構造の子セクションでは、新しいユーザがそのユーザ ドメインの管理者ロールを持つ ExampleCorp ユーザ ドメイン内に配置されるように設定されます。 新しいユーザを作成するときは、この例に示す情報より少ない情報または多い情報を指定できます。 追加属性は、後続の操作で設定できます。

（`?rsp-subtree=modified` を POST URI に追加することで）応答が要求された場合は、成功した操作により次の応答本文が返されます。

```
{
  "imdata" : [{
      "aaaUser" : {
        "attributes" : {
          "instanceId" : "0:0",
          "accountStatus" : "active",
          "childAction" : "deleteNonPresent",
          "clearPwdHistory" : "no",
          "descr" : "",
          "dn" : "uni/userext/user-georgewa",
          "email" : "georgewa@example.com",
          "encPwd" : "",
          "expiration" : "never",
          "expires" : "no",
          "firstName" : "George",
          "intId" : "none",
          "lastName" : "Washington",
          "lcOwn" : "local",
          "name" : "georgewa",
          "phone" : "5551212",
          "pwd" : "password1",
          "pwdLifeTime" : "no-password-expire",
          "pwdSet" : "no",
          "replTs" : "never",
          "rn" : "",
          "status" : "created"
        },
        "children" : [{
            "aaaUserDomain" : {
              "attributes" : {
                "instanceId" : "0:0",
                "childAction" : "deleteNonPresent",
                "descr" : "",
                "dn" : "",
                "intId" : "none",
                "lcOwn" : "local",
                "name" : "common",
                "replTs" : "never",
                "rn" : "userdomain-common",
                "status" : "created"
              },
              "children" : [{
                  "aaaUserRole" : {
                    "attributes" : {
                      "instanceId" : "0:0",
```

```
                        "childAction" : "deleteNonPresent",
                        "descr" : "",
                        "dn" : "",
                        "intId" : "none",
                        "lcOwn" : "local",
                        "name" : "read-only",
                        "privType" : "readPriv",
                        "replTs" : "never",
                        "rn" : "role-read-only",
                        "status" : "created"
                      }
                    }
                  }
                ]
              }
            }, {
              "aaaUserDomain" : {
                "attributes" : {
                  "instanceId" : "0:0",
                  "childAction" : "deleteNonPresent",
                  "descr" : "",
                  "dn" : "",
                  "intId" : "none",
                  "lcOwn" : "local",
                  "name" : "ExampleCorp",
                  "replTs" : "never",
                  "rn" : "userdomain-ExampleCorp",
                  "status" : "created"
                },
                "children" : [{
                    "aaaUserRole" : {
                      "attributes" : {
                        "instanceId" : "0:0",
                        "childAction" : "deleteNonPresent",
                        "descr" : "",
                        "dn" : "",
                        "intId" : "none",
                        "lcOwn" : "local",
                        "name" : "admin",
                        "privType" : "writePriv",
                        "replTs" : "never",
                        "rn" : "role-admin",
                        "status" : "created"
                      }
                    }
                  }
                ]
              }
            }
          ]
        }
      }
    ]
}
```

この応答は、ユーザが 2 つのユーザ ドメインに作成され追加されたことを示します。 要求通りに、ユーザには ExampleCorp ユーザ ドメインでの管理者権限があります。 デフォルトでは、ユーザには共通のユーザ ドメインでの読み取り専用権限もあります。

ユーザを削除するには、次の HTTP DELETE メッセージを送信します。

```
DELETE https://192.0.20.123/api/mo/uni/userext/user-georgewa.json
```

または、次の例のように、aaa:User 属性内に十分な命名情報と `"status":"deleted"` が含まれる HTTP POST メッセージを送信できます。

```
POST https://192.0.20.123/api/mo.json
```

```
{
  "aaaUser" : {
    "attributes" : {
      "dn" : "uni/userext/user-georgewa",
      "status" : "deleted"
    }
  }
}
```

例：cURL による JSON API を使用したユーザの追加

新しいオブジェクトを作成するには、URI またはメッセージの本文でクラスと十分な命名情報を指定する必要があります。 この例では、URI には DN が含まれ、メッセージの本文にはオブジェクト クラスが含まれます。

「例：JSON API を使用したユーザの追加, （10 ページ）」に示す形式で JSON 構造を含むローカル ファイルを作成します。 この例では、ファイル名は newuser.json で、そのファイルは別のディレクトリにあります。

cURL コマンドラインで、HTTP 方式（POST など）、ローカル ファイルのパスと名前、および API 操作の URI を指定する必要があります。

curl コマンドを使用して、次の例のように POST メッセージ内の JSON ファイルを API に送信します。

```
curl -X POST --data "@../users/newuser.json"
https://192.0.20.123/api/mo/uni/userext/user-georgewa.json
```

（注） ファイル名の前に@記号を必ず入力してください。cURL ツールは無料でオープン ソフトウェアです。 cURL の使用に関する詳細情報はオンラインで確認できます。

（`?rsp-subtree=modified` を POST URI に追加することで）応答が要求された場合は、成功した操作により「例：JSON API を使用したユーザの追加, （10 ページ）」に示す応答本文が返されます。

ユーザを削除するには、JSON ファイルを編集して `"status":"deleted"` の aaaUser 属性を追加して、コマンドを再度送信します。

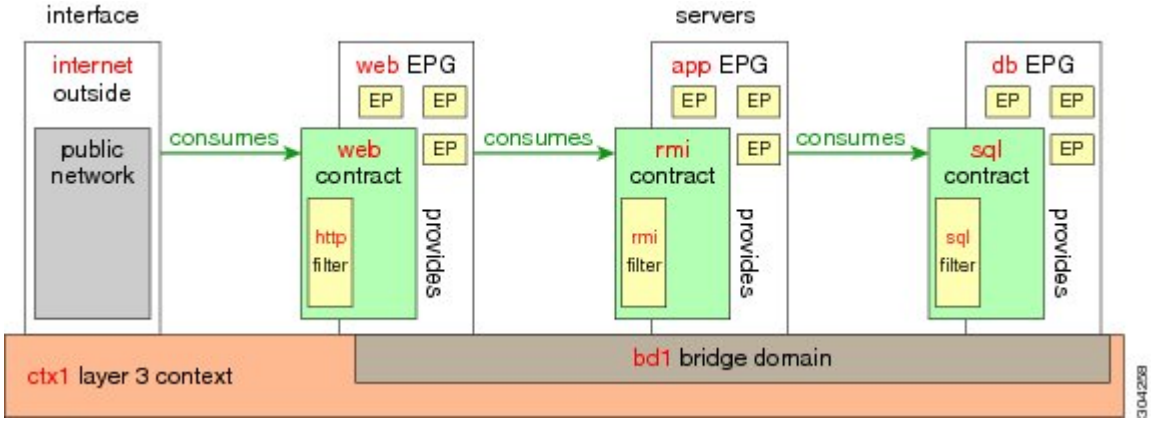
または、この HTTP DELETE メッセージを送信できます。

```
curl -X DELETE https://192.0.20.123/api/mo/uni/userext/user-georgewa.json
```

3 層アプリケーションのエンドポイント グループおよびコントラクトの導入

Cisco Application Centric Infrastructure（ACI）ファブリックの一般的な用途は、テナント ネットワーク内で 3 層アプリケーションをホストすることです。 APIC REST API を使用して、このようなア

プリケーションのエンドポイントグループとコントラクトを設定できます。この例では、アプリケーションは、次の図に示すように 3 台のサーバ、Web サーバ、アプリケーション サーバ、およびデータベース サーバを使用して実装されます。



以下は、構成の主要コンポーネントです。

- エンドポイントグループ（EPG）：各サーバは、各エンドポイントグループに関連付けられたコントラクトに従って他のエンドポイントグループからトラフィックを供給されるか消費するエンドポイント グループを表します。
- コントラクト：エンドポイントで送受信されるトラフィックはコントラクトによって抑制され、コントラクトはエンドポイントから提供されたトラフィックとエンドポイントで消費されるトラフィックのフィルタ セットを適用します。
- フィルタ：フィルタは、フィルタを含むコントラクトにより許可または拒否されるデータ プロトコルを指定します。

次のネットワーク コンポーネントは、すでに存在すると想定されています。

- ブリッジ ドメイン（BD）：レイヤ 3 コンテキストに関連付けられたレイヤ 2 ネットワーク
- コンテキスト：サーバ間の通信用のレイヤ 3 プライベート ネットワーク。
- 外部ポート：アプリケーションへの外部アクセス用のレイヤ 3 ネットワーク ポート。
- Virtual Machine Manager（VMM）およびハイパーバイザ コンポーネント
- VLAN

（注） 3 層アプリケーションの導入に関する拡張された包括的な例については、『*Cisco APIC Getting Started Guide*』を参照してください。

構成情報は、テナントの識別名（DN）に送信される単一の XML 構造に含まれています。この例では、テナントの名前は ExampleCorp です。 XML API を使用してアプリケーションを展開するには、次の HTTP POST メッセージを送信します。

```
POST https://192.0.20.123/api/mo/uni/tn-ExampleCorp.xml
```

次の XML 構造を POST メッセージの本文に含めます。

```
<fvTenant name="ExampleCorp">

    <fvAp name="OnlineStore">
        <fvAEPg name="web">
            <fvRsBd tnFvBDName="bd1"/>
            <fvRsCons tnVzBrCPName="rmi"/>
            <fvRsProv tnVzBrCPName="web"/>
            <fvRsDomAtt tDn="uni/vmmp-VMware/dom-datacenter"/>
        </fvAEPg>

        <fvAEPg name="db">
            <fvRsBd tnFvBDName="bd1"/>
            <fvRsProv tnVzBrCPName="sql"/>
            <fvRsDomAtt tDn="uni/vmmp-VMware/dom-datacenter"/>
        </fvAEPg>

        <fvAEPg name="app">
            <fvRsBd tnFvBDName="bd1"/>
            <fvRsProv tnVzBrCPName="rmi"/>
            <fvRsCons tnVzBrCPName="sql"/>
            <fvRsDomAtt tDn="uni/vmmp-VMware/dom-datacenter"/>
        </fvAEPg>
    </fvAp>

    <vzFilter name="http" >
        <vzEntry dFromPort="80" name="DPort-80" prot="tcp" etherT="ip" />
        <vzEntry dFromPort="443" name="DPort-443" prot="tcp" etherT="ip" />
    </vzFilter>

    <vzFilter name="rmi" >
        <vzEntry dFromPort="1099" name="DPort-1099" prot="tcp" etherT="ip" />
    </vzFilter>

    <vzFilter name="sql">
        <vzEntry dFromPort="1521" name="DPort-1521" prot="tcp" etherT="ip" />
    </vzFilter>

    <vzBrCP name="web">
        <vzSubj name="web">
            <vzRsSubjFiltAtt tnVzFilterName="http"/>
        </vzSubj>
    </vzBrCP>

    <vzBrCP name="rmi">
        <vzSubj name="rmi">
            <vzRsSubjFiltAtt tnVzFilterName="rmi"/>
        </vzSubj>
    </vzBrCP>

    <vzBrCP name="sql">
        <vzSubj name="sql">
           <vzRsSubjFiltAtt tnVzFilterName="sql"/>
        </vzSubj>
    </vzBrCP>
</fvTenant>
```

XML 構造では、最初の行が ExampleCorp という名前のテナントを変更または必要に応じて作成します。

```
<fvTenant name="ExampleCorp">
```

次の行は、OnlineStore という名前のアプリケーション ネットワーク プロファイルを作成します。

```
<fvAp name="OnlineStore">
```

アプリケーション ネットワーク プロファイル内の要素は、3 つのエンドポイント グループを作成します（3 台のサーバそれぞれに 1 つずつ）。 次の行は、web という名前のエンドポイント グループを作成し、bd1 という名前の既存のブリッジ ドメインに関連付けます。 このエンドポイント グループは、rmi という名前のバイナリ コントラクトで許可されたトラフィックのコンシューマまたは宛先であり、web という名前のバイナリ コントラクトで許可されたトラフィックのプロバイダーまたは送信元です。 エンドポイント グループは、datacenter という名前の VMM ドメインに関連付けられます。

```
<fvAEPg name="web">
    <fvRsBd tnFvBDName="bd1"/>
    <fvRsCons tnVzBrCPName="rmi"/>
    <fvRsProv tnVzBrCPName="web"/>
    <fvRsDomAtt tDn="uni/vmmp-VMware/dom-datacenter"/>
</fvAEPg>
```

残りの 2 つのエンドポイント グループは、アプリケーション サーバとデータベース サーバに対し、同様の方法で作成されます。

次の行は、TCP トラフィックのタイプ HTTP（ポート 80）および HTTPS（ポート 443）を指定する http という名前のトラフィック フィルタを定義します。

```
<vzFilter name="http">
    <vzEntry dFromPort="80" name="DPort-80" prot="tcp"  />
    <vzEntry dFromPort="443" name="DPort-443" prot="tcp"  />
</vzFilter>
```

残りの2つのフィルタは、アプリケーションのデータおよびデータベース（sql）のデータに対し、同様の方法で作成されます。

次の行は、http という名前のフィルタを組み込む web という名前のバイナリ コントラクトを作成します。

```
<vzBrCP name="web">
    <vzSubj name="web">
        <vzRsSubjFiltAtt tnVzFilterName="http"/>
    </vzSubj>
</vzBrCP>
```

残りの 2 つのコントラクトは、rmi および sql のデータ プロトコルに対し、同様の方法で作成されます。

最後の行は、構造を閉じます。

```
</fvTenant>
```