CHAPTER 3

VNMC XML API Methods

This section includes the following topics:

Unsupported Methods

Cisco does not support the following methods even though they appear in the Methods list on your VNMC server.

Caution We strongly recommend that you do not use the following methods. They can cause unexpected results on your VNMC server.

- aaaCheckComputeAuthToken
- aaaCheckComputeExtAccess
- aaaGetComputeAuthToken
- cliviewConfMos
- configFindDependencies
- configMoChangeEvent
- fsmDebugAction
- methodVessel
- orgResolveLogicalParents
- policyEstimateImpact
- statsClearInterval
- syntheticFSObjInventory
- syntheticTestTx

Supported Methods

This section provides descriptions, request and response syntax, and usage examples for the following VNMC API methods:

- aaaGetRemoteUserRoles
- aaaGetUserLocales
- aaaKeepAlive
- aaaLogin
- aaaLogout
- aaaRefresh
- configConfFiltered
- configConfMo
- configConfMoGroup
- configConfMos
- configFindDnsByClassId
- configResolveChildren
- configResolveClass
- configResolveClasses
- configResolveDn
- configResolveDns
- configResolveParent
- configScope
- eventSendHeartbeat
- eventSubscribe
- eventSubscribeApps
- faultAckFault
- faultAckFaults
- faultResolveFault
- loggingSyncOcns
- orgResolveElements
- orgResolveInScope
- poolResolveInScope

These methods are also called from the GUI console.

aaaGetRemoteUserRoles

This API method returns user privileges for the remote location.

Request Syntax

```
<xs:element name="aaaGetRemoteUserRoles" type="aaaGetRemoteUserRoles"
substitutionGroup="externalMethod"/>
        <xs:complexType name="aaaGetRemoteUserRoles" mixed="true">
            <xs:attribute name="inRemoteUserName">
                <xs:simpleType>
                    <xs:restriction base="xs:string">
                         <xs:pattern value="[a-zA-Z][a-zA-Z0-9_.@-]{0,31}"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:attribute>
            <xs:attribute name="cookie" type="xs:string"/>
            <xs:attribute name="response" type="YesOrNo"/>
        </xs:complexType>
```

Response Syntax

```
<xs:element name="aaaGetRemoteUserRoles" type="aaaGetRemoteUserRoles"
substitutionGroup="externalMethod"/>
        <xs:complexType name="aaaGetRemoteUserRoles" mixed="true">
            <xs:attribute name="outRemoteUserPriv">
            <xs:simpleType>
                <xs:restriction base="xs:string">
                    <xs:pattern
value="((policy|aaa|read-only|admin|tenant|operations|res-config|fault),){0,7}(policy|aaa|
read-only|admin|tenant|operations|res-config|fault){0,1}"/>
```

```
            </xs:restriction>
        </xs:simpleType>
        </xs:attribute>
        <xs:attribute name="cookie" type="xs:string"/>
        <xs:attribute name="response" type="YesOrNo"/>
        <xs:attribute name="errorCode" type="xs:unsignedInt"/>
        <xs:attribute name="errorDescr" type="xs:string"/>
        <xs:attribute name="invocationResult" type="xs:string"/>
    </xs:complexType>
```

Example

Request

```
<aaaGetRemoteUserRoles
    cookie="<real_cookie>"
    inRemoteUserName="adminuser"
    outRemoteUserPriv/>
```

Response

```
<aaaGetRemoteUserRoles
    cookie="<real_cookie>"
    commCookie="11/15/0/2964"
    srcExtSys="10.193.33.109"
    destExtSys="10.193.33.109"
    srcSvc="sam_extXMLApi"
    destSvc="mgmt-controller_dme"
    response="yes"
    outRemoteUserPriv="admin">
</aaaGetRemoteUserRoles>
```

aaaGetUserLocales

This API method returns a list of authorized user locales.

Request Syntax

```
<xs:element name="aaaGetUserLocales" type="aaaGetUserLocales"
substitutionGroup="externalMethod"/>
        <xs:complexType name="aaaGetUserLocales" mixed="true">
            <xs:attribute name="inUserName">
                <xs:simpleType>
                    <xs:restriction base="xs:string">
                        <xs:pattern value="[a-zA-Z][a-zA-Z0-9_.@-]{0,31}"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:attribute>
            <xs:attribute name="inIsUserRemote">
                <xs:simpleType>
                    <xs:union memberTypes="xs:boolean">
                        <xs:simpleType>
                            <xs:restriction base="xs:string">
                                <xs:enumeration value="no"/>
                                <xs:enumeration value="yes"/>
                            </xs:restriction>
                        </xs:simpleType>
```

```
                    </xs:union>
                </xs:simpleType>
            </xs:attribute>
            <xs:attribute name="cookie" type="xs:string"/>
            <xs:attribute name="response" type="YesOrNo"/>
        </xs:complexType>
```

Response Syntax

```
<xs:element name="aaaGetUserLocales" type="aaaGetUserLocales"
substitutionGroup="externalMethod"/>
        <xs:complexType name="aaaGetUserLocales" mixed="true">
            <xs:attribute name="outUserLocales">
                <xs:simpleType>
                    <xs:restriction base="xs:string">
                         <xs:minLength value="0"/>
                         <xs:maxLength value="512"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:attribute>
            <xs:attribute name="cookie" type="xs:string"/>
            <xs:attribute name="response" type="YesOrNo"/>
            <xs:attribute name="errorCode" type="xs:unsignedInt"/>
            <xs:attribute name="errorDescr" type="xs:string"/>
            <xs:attribute name="invocationResult" type="xs:string"/>
        </xs:complexType>
```

Example

Request

```
<aaaGetUserLocales
    cookie="<real_cookie>"
    inUserName="john"
    inIsUserRemote="no"
    outUserLocales/>
```

Response

```
<aaaGetUserLocales
    cookie="<real_cookie>"
    commCookie="11/15/0/2962"
    srcExtSys="10.193.33.109"
    destExtSys="10.193.33.109"
    srcSvc="sam_extXMLApi"
    destSvc="mgmt-controller_dme"
    response="yes"
    outUserLocales="TestSanity">
</aaaGetUserLocales>
```

aaaKeepAlive

The following example keeps the session active until the default session time expires and uses the same cookie after the method call.

Request Syntax

```
<xs:element name="aaaKeepAlive" type="aaaKeepAlive" substitutionGroup="externalMethod"/>
        <xs:complexType name="aaaKeepAlive" mixed="true">
            <xs:attribute name="cookie" type="xs:string"/>
            <xs:attribute name="response" type="YesOrNo"/>
        </xs:complexType>
```

Response Syntax

```
<xs:element name="aaaKeepAlive" type="aaaKeepAlive" substitutionGroup="externalMethod"/>
        <xs:complexType name="aaaKeepAlive" mixed="true">
            <xs:attribute name="cookie" type="xs:string"/>
            <xs:attribute name="response" type="YesOrNo"/>
            <xs:attribute name="errorCode" type="xs:unsignedInt"/>
            <xs:attribute name="errorDescr" type="xs:string"/>
            <xs:attribute name="invocationResult" type="xs:string"/>
        </xs:complexType>
```

Example

Request

```
<aaaKeepAlive
    cookie="<real_cookie>" />
```

Response

```
<aaaKeepAlive
    cookie="<real_cookie>"
    commCookie="11/15/0/2969"
    srcExtSys="10.193.33.109"
    destExtSys="10.193.33.109"
    srcSvc="sam_extXMLApi"
    destSvc="mgmt-controller_dme"
    response="yes">
</aaaKeepAlive>
```

aaaLogin

The following example shows the login process that is required to begin a session and which establishes an authenticated HTTPS session between the client and VNMC.

Request Syntax

```
<xs:element name="aaaLogin" type="aaaLogin" substitutionGroup="externalMethod"/>
        <xs:complexType name="aaaLogin" mixed="true">
            <xs:attribute name="inName">
                <xs:simpleType>
```

```
                    <xs:restriction base="xs:string">
                         <xs:pattern value="[\-\.:_a-zA-Z0-9]{0,16}"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:attribute>
            <xs:attribute name="inPassword">
                <xs:simpleType>
                    <xs:restriction base="xs:string">
                         <xs:minLength value="0"/>
                         <xs:maxLength value="512"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:attribute>
            <xs:attribute name="cookie" type="xs:string"/>
            <xs:attribute name="response" type="YesOrNo"/>
        </xs:complexType>
```

Response Syntax

```
<xs:element name="aaaLogin" type="aaaLogin" substitutionGroup="externalMethod"/>
        <xs:complexType name="aaaLogin" mixed="true">
            <xs:attribute name="outCookie" type="xs:string"/>
            <xs:attribute name="outRefreshPeriod" type="xs:unsignedInt"/>
            <xs:attribute name="outPriv">
            <xs:simpleType>
                <xs:restriction base="xs:string">
                    <xs:pattern
value="((policy|aaa|read-only|admin|tenant|operations|res-config|fault),){0,7}(policy|aaa|
read-only|admin|tenant|operations|res-config|fault){0,1}"/>
                </xs:restriction>
            </xs:simpleType>
            </xs:attribute>
            <xs:attribute name="outDomains" type="xs:string"/>
            <xs:attribute name="outChannel">
                <xs:simpleType>
                    <xs:restriction base="xs:string">
                         <xs:enumeration value="fullssl"/>
                         <xs:enumeration value="noencssl"/>
                         <xs:enumeration value="plain"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:attribute>
            <xs:attribute name="outEvtChannel">
                <xs:simpleType>
                    <xs:restriction base="xs:string">
                         <xs:enumeration value="fullssl"/>
                         <xs:enumeration value="noencssl"/>
                         <xs:enumeration value="plain"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:attribute>
            <xs:attribute name="outSessionId">
                <xs:simpleType>
                    <xs:restriction base="xs:string">
                         <xs:minLength value="1"/>
                         <xs:maxLength value="64"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:attribute>
            <xs:attribute name="outVersion" type="xs:string"/>
            <xs:attribute name="cookie" type="xs:string"/>
            <xs:attribute name="response" type="YesOrNo"/>
```

```
            <xs:attribute name="errorCode" type="xs:unsignedInt"/>
            <xs:attribute name="errorDescr" type="xs:string"/>
            <xs:attribute name="invocationResult" type="xs:string"/>
        </xs:complexType>
```

Example

Request

```
<aaaLogin
    inName="admin"
    inPassword="Nbv12345"/>
```

Response

```
<aaaLogin cookie=""
    commCookie=""
    srcExtSys="0.0.0.0"
    destExtSys="0.0.0.0"
    srcSvc="" destSvc=""
    response="yes"
    outCookie="<real_cookie>"
    outRefreshPeriod="600"
    outPriv="admin"
    outDomains=""
    outChannel="fullssl"
    outEvtChannel="fullssl"
    outSessionId="web_49019"
    outVersion="1.0(0.39938)">
</aaaLogin>
```

aaaLogout

The following example shows the logout process to end a current session. When the default session time period expires, the process is called automatically.

Request Syntax

```
<xs:element name="aaaLogout" type="aaaLogout" substitutionGroup="externalMethod"/>
        <xs:complexType name="aaaLogout" mixed="true">
            <xs:attribute name="inCookie" type="xs:string"/>
            <xs:attribute name="cookie" type="xs:string"/>
            <xs:attribute name="response" type="YesOrNo"/>
        </xs:complexType>
```

Response Syntax

```
<xs:element name="aaaLogout" type="aaaLogout" substitutionGroup="externalMethod"/>
        <xs:complexType name="aaaLogout" mixed="true">
            <xs:attribute name="outStatus">
                <xs:simpleType>
                    <xs:restriction base="xs:string">
                         <xs:enumeration value="success"/>
                         <xs:enumeration value="failure"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:attribute>
```

```
            <xs:attribute name="cookie" type="xs:string"/>
            <xs:attribute name="response" type="YesOrNo"/>
            <xs:attribute name="errorCode" type="xs:unsignedInt"/>
            <xs:attribute name="errorDescr" type="xs:string"/>
            <xs:attribute name="invocationResult" type="xs:string"/>
        </xs:complexType>
```

Example

Request

```
<aaaLogout
    inCookie="<real_cookie>"
    outStatus/>
```

Response

```
<aaaLogout cookie=""
    commCookie=""
    srcExtSys="0.0.0.0"
    destExtSys="0.0.0.0"
    srcSvc=""
    destSvc=""
    response="yes"
    outStatus="success">
</aaaLogout>
```

aaaRefresh

Sessions can be kept active (within the default session time frame) by user activity. There is a default of 7200 seconds that counts down when inactivity begins. If the 7200 seconds expire, VNMC enters a sleep mode and requires signing back in, which restarts the countdown. The session continues using the same session ID.

Note Using this method expires the previous cookie and issues a new cookie.

Request Syntax

```
<xs:element name="aaaRefresh" type="aaaRefresh" substitutionGroup="externalMethod"/>
        <xs:complexType name="aaaRefresh" mixed="true">
            <xs:attribute name="inName">
                <xs:simpleType>
                    <xs:restriction base="xs:string">
                        <xs:pattern value="[\-\.:_a-zA-Z0-9]{0,16}"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:attribute>
            <xs:attribute name="inPassword">
                <xs:simpleType>
                    <xs:restriction base="xs:string">
                        <xs:minLength value="0"/>
                        <xs:maxLength value="512"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:attribute>
            <xs:attribute name="inCookie" type="xs:string"/>
```

```
            <xs:attribute name="cookie" type="xs:string"/>
            <xs:attribute name="response" type="YesOrNo"/>
        </xs:complexType>
```

Response Syntax

```
<xs:element name="aaaRefresh" type="aaaRefresh" substitutionGroup="externalMethod"/>
        <xs:complexType name="aaaRefresh" mixed="true">
            <xs:attribute name="outCookie" type="xs:string"/>
            <xs:attribute name="outRefreshPeriod" type="xs:unsignedInt"/>
            <xs:attribute name="outPriv">
            <xs:simpleType>
                <xs:restriction base="xs:string">
                    <xs:pattern
value="((policy|aaa|read-only|admin|tenant|operations|res-config|fault),){0,7}(policy|aaa|
read-only|admin|tenant|operations|res-config|fault){0,1}"/>
                </xs:restriction>
            </xs:simpleType>
            </xs:attribute>
            <xs:attribute name="outDomains" type="xs:string"/>
            <xs:attribute name="outChannel">
                <xs:simpleType>
                    <xs:restriction base="xs:string">
                         <xs:enumeration value="fullssl"/>
                         <xs:enumeration value="noencssl"/>
                         <xs:enumeration value="plain"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:attribute>
            <xs:attribute name="outEvtChannel">
                <xs:simpleType>
                    <xs:restriction base="xs:string">
                         <xs:enumeration value="fullssl"/>
                         <xs:enumeration value="noencssl"/>
                         <xs:enumeration value="plain"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:attribute>
            <xs:attribute name="cookie" type="xs:string"/>
            <xs:attribute name="response" type="YesOrNo"/>
            <xs:attribute name="errorCode" type="xs:unsignedInt"/>
            <xs:attribute name="errorDescr" type="xs:string"/>
            <xs:attribute name="invocationResult" type="xs:string"/>
        </xs:complexType>
```

Example

Request

```
<aaaRefresh
    cookie="<real_cookie>"
    inName="admin"
    inPassword="Nbv12345"
    inCookie="<real_cookie>"/>
```

Response

```
<aaaRefresh
    cookie="<real_cookie>"
    commCookie="" srcExtSys="0.0.0.0"
    destExtSys="0.0.0.0"
    srcSvc=""
    destSvc=""
    response="yes"
    outCookie="<real_cookie>"
    outRefreshPeriod="600"
    outPriv="admin"
    outDomains=""
    outChannel="fullssl"
    outEvtChannel="fullssl">
</aaaRefresh>
```

configConfFiltered

The following example shows how data and activity are limited according to the configured policies.

Request Syntax

```
<xs:element name="configConfFiltered" type="configConfFiltered"
substitutionGroup="externalMethod"/>
        <xs:complexType name="configConfFiltered" mixed="true">
            <xs:all>
                <xs:element name="inFilter" type="filterFilter" minOccurs="0"/>
                <xs:element name="inConfig" type="configConfig" minOccurs="0"/>
            </xs:all>
            <xs:attribute name="inHierarchical">
                <xs:simpleType>
                    <xs:union memberTypes="xs:boolean">
                         <xs:simpleType>
                             <xs:restriction base="xs:string">
                                 <xs:enumeration value="no"/>
                                 <xs:enumeration value="yes"/>
                             </xs:restriction>
                         </xs:simpleType>
                    </xs:union>
                </xs:simpleType>
            </xs:attribute>
            <xs:attribute name="cookie" type="xs:string"/>
            <xs:attribute name="response" type="YesOrNo"/>
            <xs:attribute name="classId" type="namingClassId"/>
        </xs:complexType>
```

Response Syntax

```
<xs:element name="configConfFiltered" type="configConfFiltered"
substitutionGroup="externalMethod"/>
        <xs:complexType name="configConfFiltered" mixed="true">
            <xs:all>
                <xs:element name="outConfigs" type="configSet" minOccurs="0"/>
            </xs:all>
            <xs:attribute name="cookie" type="xs:string"/>
            <xs:attribute name="response" type="YesOrNo"/>
            <xs:attribute name="errorCode" type="xs:unsignedInt"/>
            <xs:attribute name="errorDescr" type="xs:string"/>
```

```
            <xs:attribute name="invocationResult" type="xs:string"/>
            <xs:attribute name="classId" type="namingClassId"/>
        </xs:complexType>
```

Example

Request

```
<configConfFiltered
    cookie="<real_cookie>"
    inHierarchical="false"
    classId="orgTenant">
    <inFilter>
        <eq class="orgTenant"
            property="name"
            value="Tenant1" />
    </inFilter>
    <inConfig>
        <orgDatacenter
            dn="org-HR"
            descr="HR (Human Resources- new Descr)"/>
    </inConfig>
</configConfFiltered>
```

Response

```
<configConfFiltered
    cookie="<real_cookie>"
    commCookie="5/15/0/617"
    srcExtSys="10.193.33.206"
    destExtSys="10.193.33.206"
    srcSvc="sam_extXMLApi"
    destSvc="resource-mgr_dme"
    response="yes"
    classId="orgTenant">
    <outConfigs>
        <orgDatacenter
            descr="HR (Human Resources- new Descr)"
            dn="org-root/org-tenant1/org-HR"
            fltAggr="0"
            level="2"
            name="HR"
            status="modified"/>
    </outConfigs>
</configConfFiltered>
```

configConfMo

The following example shows how the specified Managed Object is configured in a single subtree (for example, DN).

Request Syntax

```
<xs:element name="configConfMo" type="configConfMo" substitutionGroup="externalMethod"/>
        <xs:complexType name="configConfMo" mixed="true">
            <xs:all>
                <xs:element name="inConfig" type="configConfig" minOccurs="0"/>
            </xs:all>
```

```
            <xs:attribute name="inHierarchical">
                <xs:simpleType>
                    <xs:union memberTypes="xs:boolean">
                         <xs:simpleType>
                             <xs:restriction base="xs:string">
                                 <xs:enumeration value="no"/>
                                 <xs:enumeration value="yes"/>
                             </xs:restriction>
                         </xs:simpleType>
                    </xs:union>
                </xs:simpleType>
            </xs:attribute>
            <xs:attribute name="cookie" type="xs:string"/>
            <xs:attribute name="response" type="YesOrNo"/>
            <xs:attribute name="dn" type="referenceObject"/>
        </xs:complexType>
```

Response Syntax

```
<xs:element name="configConfMo" type="configConfMo" substitutionGroup="externalMethod"/>
        <xs:complexType name="configConfMo" mixed="true">
            <xs:all>
                <xs:element name="outConfig" type="configConfig" minOccurs="0"/>
            </xs:all>
            <xs:attribute name="cookie" type="xs:string"/>
            <xs:attribute name="response" type="YesOrNo"/>
            <xs:attribute name="errorCode" type="xs:unsignedInt"/>
            <xs:attribute name="errorDescr" type="xs:string"/>
            <xs:attribute name="invocationResult" type="xs:string"/>
            <xs:attribute name="dn" type="referenceObject"/>
        </xs:complexType>
```

Example

Request

```
<configConfMo
    dn=""
    cookie="<real_cookie>"
    inHierarchical="false">
    <inConfig>
        <aaaLdapEp
            attribute="CiscoAvPair"
            basedn="dc=pasadena,dc=company123,dc=com"
            descr=""
            dn="sys/ldap-ext"
            filter="sAMAccountName=$userid"
            retries="1"
            status="modified"
            timeout="30"/>
    </inConfig>
</configConfMo>
```

Response

```
<configConfMo
    dn=""
    cookie="<real_cookie>"
    commCookie="11/15/0/28"
    srcExtSys="10.193.33.101"
    destExtSys="10.193.33.101"
```

```
    srcSvc="sam_extXMLApi"
    destSvc="mgmt-controller_dme"
    response="yes">
    <outConfig>
        <aaaLdapEp
            attribute="CiscoAvPair"
            basedn="dc=pasadena,dc=company123,dc=com"
            childAction="deleteNonPresent"
            descr=""
            dn="sys/ldap-ext"
            filter="sAMAccountName=$userid"
            fsmDescr=""
            fsmPrev="updateEpSuccess"
            fsmProgr="100"
            fsmStageDescr=""
            fsmStamp="2010-11-22T23:41:01.826"
            fsmStatus="nop"
            fsmTry="0"
            intId="10027"
            name=""
            retries="1"
            status="modified"
            timeout="30"/>
    </outConfig>
</configConfMo>
```

configConfMoGroup

The following example shows how groups of managed objects are configured based upon the configured policies.

Request Syntax

```
<xs:element name="configConfMoGroup" type="configConfMoGroup"
substitutionGroup="externalMethod"/>
        <xs:complexType name="configConfMoGroup" mixed="true">
            <xs:all>
                <xs:element name="inDns" type="dnSet" minOccurs="0"/>
                <xs:element name="inConfig" type="configConfig" minOccurs="0"/>
            </xs:all>
            <xs:attribute name="inHierarchical">
                <xs:simpleType>
                    <xs:union memberTypes="xs:boolean">
                        <xs:simpleType>
                            <xs:restriction base="xs:string">
                                <xs:enumeration value="no"/>
                                <xs:enumeration value="yes"/>
                            </xs:restriction>
                        </xs:simpleType>
                    </xs:union>
                </xs:simpleType>
            </xs:attribute>
            <xs:attribute name="cookie" type="xs:string"/>
            <xs:attribute name="response" type="YesOrNo"/>
        </xs:complexType>
```

Response Syntax

```
<xs:element name="configConfMoGroup" type="configConfMoGroup"
substitutionGroup="externalMethod"/>
        <xs:complexType name="configConfMoGroup" mixed="true">
            <xs:all>
                <xs:element name="outConfigs" type="configSet" minOccurs="0"/>
            </xs:all>
            <xs:attribute name="cookie" type="xs:string"/>
            <xs:attribute name="response" type="YesOrNo"/>
            <xs:attribute name="errorCode" type="xs:unsignedInt"/>
            <xs:attribute name="errorDescr" type="xs:string"/>
            <xs:attribute name="invocationResult" type="xs:string"/>
        </xs:complexType>
```

Example

Note The descr property of orgDataCenter (under org-root/org-tenant1 and org-root/org-tenant2) is modified. Because the descr property is not implicit, it can be modified. If implicit, the modification does not apply and a new orgDataCenter is created.

Request

```
<configConfMoGroup
    cookie="<real_cookie>"
    inHierarchical="false">
    <inDns>
        <dn value="org-root/org-tenant1" />
        <dn value="org-root/org-tenant2" />
    </inDns>
    <inConfig>
        <orgDatacenter
            dn="org-HR"
            descr="HR (Human Resources)"/>
    </inConfig>
</configConfMoGroup>
```

Response

```
<configConfMoGroup
    cookie="<real_cookie>"
    commCookie="5/15/0/600"
    srcExtSys="10.193.33.206"
    destExtSys="10.193.33.206"
    srcSvc="sam_extXMLApi"
    destSvc="resource-mgr_dme"
    response="yes">
    <outConfigs>
        <orgDatacenter
            descr="HR (Human Resources)"
            dn="org-root/org-Cola/org-HR"
            fltAggr="0"
            level="2"
            name="HR"
            status="modified"/>
        <orgDatacenter
            descr="HR (Human Resources)"
            dn="org-root/org-tenant1/org-HR"
            fltAggr="0"
```

```
            level="2"
            name="HR"
            status="modified"/>
    </outConfigs>
</configConfMoGroup>
```

configConfMos

The following example shows how to configure managed objects in multiple subtrees using DNs.

Request Syntax

```
<xs:element name="configConfMos" type="configConfMos" substitutionGroup="externalMethod"/>
        <xs:complexType name="configConfMos" mixed="true">
            <xs:all>
                <xs:element name="inConfigs" type="configMap" minOccurs="0">
                    <xs:unique name="unique_map_key_2">
                         <xs:selector xpath="pair"/>
                         <xs:field xpath="@key"/>
                    </xs:unique>
                </xs:element>
            </xs:all>
            <xs:attribute name="inHierarchical">
                <xs:simpleType>
                    <xs:union memberTypes="xs:boolean">
                         <xs:simpleType>
                             <xs:restriction base="xs:string">
                                 <xs:enumeration value="no"/>
                                 <xs:enumeration value="yes"/>
                             </xs:restriction>
                         </xs:simpleType>
                    </xs:union>
                </xs:simpleType>
            </xs:attribute>
            <xs:attribute name="cookie" type="xs:string"/>
            <xs:attribute name="response" type="YesOrNo"/>
        </xs:complexType>
```

Response Syntax

```
<xs:element name="configConfMos" type="configConfMos" substitutionGroup="externalMethod"/>
        <xs:complexType name="configConfMos" mixed="true">
            <xs:all>
                <xs:element name="outConfigs" type="configMap" minOccurs="0">
                    <xs:unique name="unique_map_key_4">
                         <xs:selector xpath="pair"/>
                         <xs:field xpath="@key"/>
                    </xs:unique>
                </xs:element>
            </xs:all>
            <xs:attribute name="cookie" type="xs:string"/>
            <xs:attribute name="response" type="YesOrNo"/>
            <xs:attribute name="errorCode" type="xs:unsignedInt"/>
            <xs:attribute name="errorDescr" type="xs:string"/>
            <xs:attribute name="invocationResult" type="xs:string"/>
        </xs:complexType>
```

Example

Request

```
<configConfMos
    cookie="<real_cookie>">
    <inConfigs>
        <pair key="org-root/logprof-default">
            <policyLogProfile dn="org-root/logprof-default"
                name="default"
                level="debug1"
                size="10000000"
                backupCount="4"/>
        </pair>

    <!-- Update Controller Device Profile -->
        <pair key="org-root/controller-profile-default">
            <policyControllerDeviceProfile
                dn="org-root/controller-profile-default"
                adminState="enabled">

                <commDnsProvider hostip="171.70.168.183" order="1"/>
                <commDnsProvider hostip="171.68.226.120" order="2"/>
                <commDnsProvider hostip="64.102.6.247"   order="3"/>
            </policyControllerDeviceProfile>
        </pair>
    </inConfigs>
</configConfMos>
```

Response

```
<configConfMos
    cookie="<real_cookie>"
    commCookie="7/15/0/1a74"
    srcExtSys="10.193.34.70"
    destExtSys="10.193.34.70"
    srcSvc="sam_extXMLApi"
    destSvc="policy-mgr_dme"
    response="yes">
    <outConfigs>
        <pair key="org-root/logprof-default">
            <policyLogProfile
                backupCount="4"
                descr="the log level for every process"
                dn="org-root/logprof-default"
                intId="10065"
                level="debug1"
                name="default"
                size="10000000"/>
        </pair>
        <pair key="org-root/controller-profile-default">
            <policyControllerDeviceProfile
                adminState="enabled"
                coreFilePolicy=""
                descr="default profile for management server virtual machine"
                dn="org-root/controller-profile-default"
                dnsPolicy=""
                faultPolicy="default"
                httpPolicy="default"
                httpsPolicy="default"
                intId="10057"
                logProfilePolicy="default"
                name="default"
```

```
                snmpPolicy=""
                syslogPolicy=""
                telnetPolicy=""
                timezone=""/>
        </pair>
    </outConfigs>
</configConfMos>
```

configFindDnsByClassId

The following example shows how to find distinguished names and return them sorted by class ID.

Request Syntax

```
<xs:element name="configFindDnsByClassId" type="configFindDnsByClassId"
substitutionGroup="externalMethod"/>
        <xs:complexType name="configFindDnsByClassId" mixed="true">
            <xs:all>
                <xs:element name="inFilter" type="filterFilter" minOccurs="0"/>
            </xs:all>
            <xs:attribute name="cookie" type="xs:string"/>
            <xs:attribute name="response" type="YesOrNo"/>
            <xs:attribute name="classId" type="namingClassId"/>
        </xs:complexType>
```

Response Syntax

```
<xs:element name="configFindDnsByClassId" type="configFindDnsByClassId"
substitutionGroup="externalMethod"/>
        <xs:complexType name="configFindDnsByClassId" mixed="true">
            <xs:all>
                <xs:element name="outDns" type="dnSet" minOccurs="0"/>
            </xs:all>
            <xs:attribute name="cookie" type="xs:string"/>
            <xs:attribute name="response" type="YesOrNo"/>
            <xs:attribute name="errorCode" type="xs:unsignedInt"/>
            <xs:attribute name="errorDescr" type="xs:string"/>
            <xs:attribute name="invocationResult" type="xs:string"/>
            <xs:attribute name="classId" type="namingClassId"/>
        </xs:complexType>
```

Example

Request

```
<configFindDnsByClassId
    classId="eventRecord"
    cookie="<real_cookie>" />
```

Response

```
<configFindDnsByClassId
    cookie="<real_cookie>"
    commCookie="2/12/0/1810"
    srcExtSys="172.20.101.150"
    destExtSys="172.20.101.150"
    srcSvc="sam_extXMLApi"
```

```
    destSvc="service-reg_dme"
    response="yes"
    classId="eventRecord">
    <outDns>
        <dn value="event-log/10210"/>
        <dn value="event-log/10250"/>
        <dn value="event-log/10211"/>
        <dn value="event-log/10221"/>
        <dn value="event-log/10251"/>
        <dn value="event-log/10141"/>
        <dn value="event-log/10151"/>
    </outDns>
</configFindDnsByClassId>
```

configResolveChildren

The following example shows how to retrieve children of managed objects under a specific DN in the managed information tree. A filter can be used to reduce the number of children being returned.

Request Syntax

```
<xs:element name="configResolveChildren" type="configResolveChildren"
substitutionGroup="externalMethod"/>
        <xs:complexType name="configResolveChildren" mixed="true">
            <xs:all>
                <xs:element name="inFilter" type="filterFilter" minOccurs="0"/>
            </xs:all>
            <xs:attribute name="inDn" type="referenceObject"/>
            <xs:attribute name="inHierarchical">
                <xs:simpleType>
                    <xs:union memberTypes="xs:boolean">
                         <xs:simpleType>
                             <xs:restriction base="xs:string">
                                 <xs:enumeration value="no"/>
                                 <xs:enumeration value="yes"/>
                             </xs:restriction>
                         </xs:simpleType>
                    </xs:union>
                </xs:simpleType>
            </xs:attribute>
            <xs:attribute name="cookie" type="xs:string"/>
            <xs:attribute name="response" type="YesOrNo"/>
            <xs:attribute name="classId" type="namingClassId"/>
        </xs:complexType>
```

Response Syntax

```
<xs:element name="configResolveChildren" type="configResolveChildren"
substitutionGroup="externalMethod"/>
        <xs:complexType name="configResolveChildren" mixed="true">
            <xs:all>
                <xs:element name="outConfigs" type="configSet" minOccurs="0"/>
            </xs:all>
            <xs:attribute name="cookie" type="xs:string"/>
            <xs:attribute name="response" type="YesOrNo"/>
            <xs:attribute name="errorCode" type="xs:unsignedInt"/>
```

```
            <xs:attribute name="errorDescr" type="xs:string"/>
            <xs:attribute name="invocationResult" type="xs:string"/>
            <xs:attribute name="classId" type="namingClassId"/>
        </xs:complexType>
```

Example

Request

```
<configResolveChildren
    cookie="<real_cookie>"
    classId="aaaUser"
    inDn="sys/user-ext"
    inHierarchical="false">
    <inFilter>
    </inFilter>
</configResolveChildren>
```

Response

```
<configResolveChildren
    cookie="<real_cookie>"
    commCookie="11/15/0/2a59"
    srcExtSys="10.193.33.120"
    destExtSys="10.193.33.120"
    srcSvc="sam_extXMLApi"
    destSvc="mgmt-controller_dme"
    response="yes"
    classId="aaaUser">
    <outConfigs>
        <aaaUser descr="" dn="sys/user-ext/user-doe"
            email="" expiration="never" expires="no" firstName="John" intId="12999"
            lastName="Doe" name="doe" phone="" priv="admin,read-only" pwdSet="yes"/>
        <aaaUser descr="" dn="sys/user-ext/user-jacks" email="" expiration="never"
            expires="no" firstName="Play" intId="12734" lastName="Jacks" name="jacks"
            phone="" priv="fault,operations,policy,read-only,res-config,tenant"
            pwdSet="yes"/>
        <aaaUser descr="" dn="sys/user-ext/user-admin" email="" expiration="never"
            expires="no" firstName="" intId="10052" lastName="" name="admin" phone=""
            priv="admin,read-only" pwdSet="yes"/>
        <aaaUser descr="" dn="sys/user-ext/user-over" email="" expiration="never"
            expires="no" firstName="Roll" intId="12711" lastName="Over" name="over"
            phone="" priv="fault,operations,policy,read-only,res-config,tenant"
            pwdSet="yes"/>
        <aaaUser descr="" dn="sys/user-ext/user-fun" email="" expiration="never"
            expires="no" firstName="Have" intId="12708" lastName="Fun" name="fun" phone=""
            priv="read-only" pwdSet="yes"/>
        <aaaUser descr="testuser" dn="sys/user-ext/user-aaa" email="" expiration="never"
            expires="no" firstName="a" intId="10620" lastName="aa" name="aaa" phone=""
            priv="aaa,read-only" pwdSet="no"/>
    </outConfigs>
</configResolveChildren>
```

configResolveClass

The following example shows how to return the requested managed object in a given class. If *inHierarchical* = true, the returned object will also contain its children.

Request Syntax

```
<xs:element name="configResolveClass" type="configResolveClass"
substitutionGroup="externalMethod"/>
        <xs:complexType name="configResolveClass" mixed="true">
            <xs:all>
                <xs:element name="inFilter" type="filterFilter" minOccurs="0"/>
            </xs:all>
            <xs:attribute name="inHierarchical">
                <xs:simpleType>
                    <xs:union memberTypes="xs:boolean">
                         <xs:simpleType>
                             <xs:restriction base="xs:string">
                                 <xs:enumeration value="no"/>
                                 <xs:enumeration value="yes"/>
                             </xs:restriction>
                         </xs:simpleType>
                    </xs:union>
                </xs:simpleType>
            </xs:attribute>
            <xs:attribute name="cookie" type="xs:string"/>
            <xs:attribute name="response" type="YesOrNo"/>
            <xs:attribute name="classId" type="namingClassId"/>
        </xs:complexType>
```

Response Syntax

```
<xs:element name="configResolveClass" type="configResolveClass"
substitutionGroup="externalMethod"/>
        <xs:complexType name="configResolveClass" mixed="true">
            <xs:all>
                <xs:element name="outConfigs" type="configSet" minOccurs="0"/>
            </xs:all>
            <xs:attribute name="cookie" type="xs:string"/>
            <xs:attribute name="response" type="YesOrNo"/>
            <xs:attribute name="errorCode" type="xs:unsignedInt"/>
            <xs:attribute name="errorDescr" type="xs:string"/>
            <xs:attribute name="invocationResult" type="xs:string"/>
            <xs:attribute name="classId" type="namingClassId"/>
        </xs:complexType>
```

Example

Request

```
<configResolveClass
    cookie="<real_cookie>"
    classId="pkiEp"
    inHierarchical="false">
    <inFilter>
    </inFilter>
</configResolveClass>
```

Response

```
<configResolveClass
    cookie="<real_cookie>"
    commCookie="11/15/0/2a5b"
    srcExtSys="10.193.33.120"
    destExtSys="10.193.33.120"
    srcSvc="sam_extXMLApi"
    destSvc="mgmt-controller_dme"
    response="yes"
    classId="pkiEp">
    <outConfigs>
        <pkiEp descr=""
            dn="sys/pki-ext"
            intId="10037"
            name=""/>
    </outConfigs>
</configResolveClass>
```

configResolveClasses

The following example shows how to return requested managed objects in several classes. If *inHierarchical* = true, the returned object will also contain its children.

Request Syntax

```
<xs:element name="configResolveClasses" type="configResolveClasses"
substitutionGroup="externalMethod"/>
        <xs:complexType name="configResolveClasses" mixed="true">
            <xs:all>
                <xs:element name="inIds" type="classIdSet" minOccurs="0"/>
            </xs:all>
            <xs:attribute name="inHierarchical">
                <xs:simpleType>
                    <xs:union memberTypes="xs:boolean">
                         <xs:simpleType>
                             <xs:restriction base="xs:string">
                                 <xs:enumeration value="no"/>
                                 <xs:enumeration value="yes"/>
                             </xs:restriction>
                         </xs:simpleType>
                    </xs:union>
                </xs:simpleType>
            </xs:attribute>
            <xs:attribute name="cookie" type="xs:string"/>
            <xs:attribute name="response" type="YesOrNo"/>
        </xs:complexType>
```

Response Syntax

```
<xs:element name="configResolveClasses" type="configResolveClasses"
substitutionGroup="externalMethod"/>
        <xs:complexType name="configResolveClasses" mixed="true">
            <xs:all>
                <xs:element name="outConfigs" type="configSet" minOccurs="0"/>
            </xs:all>
            <xs:attribute name="cookie" type="xs:string"/>
            <xs:attribute name="response" type="YesOrNo"/>
            <xs:attribute name="errorCode" type="xs:unsignedInt"/>
```

```
            <xs:attribute name="errorDescr" type="xs:string"/>
            <xs:attribute name="invocationResult" type="xs:string"/>
        </xs:complexType>
```

Example

Request

```
<configResolveClasses
    cookie="<real_cookie>"
    inHierarchical="false">
    <inIds>
        <Id value="eventRecord"/>
        <Id value="faultInst"/>
    </inIds>
</configResolveClasses>
```

Response

```
<configResolveClasses
    cookie="<real_cookie>"
    commCookie="2/12/0/181a"
    srcExtSys="172.20.101.150"
    destExtSys="172.20.101.150"
    srcSvc="sam_extXMLApi"
    destSvc="service-reg_dme"
    response="yes">
    <outConfigs>
        <eventRecord
            affected="observe/observed-1001-1"
            cause="transition"
            changeSet=""
            code="E4194388"
            created="2012-07-25T00:39:35.528"
            descr="[FSM:BEGIN]: Resolve Mgmt Controller
Fsm(FSM:sam:dme:ObserveObservedResolveControllerFsm)"
            dn="event-log/10133"
            id="10133"
            ind="state-transition"
            severity="info"
            trig="special"
            txId="3"
            user="internal"/>
        <eventRecord
            affected="observe/observed-1001-1"
            cause="transition"
            changeSet=""
            code="E4194388"
            created="2012-07-25T00:39:35.528"
            descr="[FSM:STAGE:END]:
(FSM-STAGE:sam:dme:ObserveObservedResolveControllerFsm:begin)"
            dn="event-log/10134"
            id="10134"
            ind="state-transition"
            severity="info"
            trig="special"
            txId="3"
            user="internal"/>
    </outConfigs>
</configResolveClasses>
```

configResolveDn

The following example shows how to retrieve a single managed object for a specified DN.

Request Syntax

```
<xs:element name="configResolveDn" type="configResolveDn"
substitutionGroup="externalMethod"/>
        <xs:complexType name="configResolveDn" mixed="true">
            <xs:attribute name="inHierarchical">
                <xs:simpleType>
                    <xs:union memberTypes="xs:boolean">
                        <xs:simpleType>
                            <xs:restriction base="xs:string">
                                <xs:enumeration value="no"/>
                                <xs:enumeration value="yes"/>
                            </xs:restriction>
                        </xs:simpleType>
                    </xs:union>
                </xs:simpleType>
            </xs:attribute>
            <xs:attribute name="cookie" type="xs:string"/>
            <xs:attribute name="response" type="YesOrNo"/>
            <xs:attribute name="dn" type="referenceObject"/>
        </xs:complexType>
```

Response Syntax

```
<xs:element name="configResolveDn" type="configResolveDn"
substitutionGroup="externalMethod"/>
        <xs:complexType name="configResolveDn" mixed="true">
            <xs:all>
                <xs:element name="outConfig" type="configConfig" minOccurs="0"/>
            </xs:all>
            <xs:attribute name="cookie" type="xs:string"/>
            <xs:attribute name="response" type="YesOrNo"/>
            <xs:attribute name="errorCode" type="xs:unsignedInt"/>
            <xs:attribute name="errorDescr" type="xs:string"/>
            <xs:attribute name="invocationResult" type="xs:string"/>
            <xs:attribute name="dn" type="referenceObject"/>
        </xs:complexType>
```

Example

Request

```
<configResolveDn
    cookie="<real_cookie>"
    dn="vmmEp/vm-mgr-vcenter1" />
```

Response

```
<configResolveDn dn="vmmEp/vm-mgr-vcenter1"
    cookie="<real_cookie>"
    commCookie="9/15/0/1c0d"
    srcExtSys="10.193.34.70"
    destExtSys="10.193.34.70"
    srcSvc="sam_extXMLApi"
```

```
    destSvc="vm-mgr_dme"
    response="yes">
    <outConfig>
        <vmManager
            adminState="enable"
            descr=""
            dn="vmmEp/vm-mgr-vcenter1"
            fltAggr="0"
            fsmDescr="AG registration with
vCenter(FSM:sam:dme:VmManagerRegisterWithVCenter)"
            fsmPrev="RegisterWithVCenterRegistering"
            fsmProgr="13"
            fsmRmtInvErrCode="none"
            fsmRmtInvErrDescr=""
            fsmRmtInvRslt=""
            fsmStageDescr="AG registration with
vCenter(FSM-STAGE:sam:dme:VmManagerRegisterWithVCenter:Registering)"
            fsmStamp="2010-11-11T21:37:15.696"
            fsmStatus="RegisterWithVCenterRegistering"
            fsmTry="1"
            hostName="vpod-31.host123.com"
            intId="21959"
            name="vcenter1"
            operState="unknown"
            stateQual=""
            type="vmware"
            version=""/>
    </outConfig>
</configResolveDn>
```

configResolveDns

The following example shows how to retrieve managed objects for a list of DNs.

Request Syntax

```
<xs:element name="configResolveDns" type="configResolveDns"
substitutionGroup="externalMethod"/>
        <xs:complexType name="configResolveDns" mixed="true">
            <xs:all>
                <xs:element name="inDns" type="dnSet" minOccurs="0"/>
            </xs:all>
            <xs:attribute name="inHierarchical">
                <xs:simpleType>
                    <xs:union memberTypes="xs:boolean">
                         <xs:simpleType>
                             <xs:restriction base="xs:string">
                                 <xs:enumeration value="no"/>
                                 <xs:enumeration value="yes"/>
                             </xs:restriction>
                         </xs:simpleType>
                    </xs:union>
                </xs:simpleType>
            </xs:attribute>
            <xs:attribute name="cookie" type="xs:string"/>
            <xs:attribute name="response" type="YesOrNo"/>
        </xs:complexType>
```

Response Syntax

```
<xs:element name="configResolveDns" type="configResolveDns"
substitutionGroup="externalMethod"/>
        <xs:complexType name="configResolveDns" mixed="true">
            <xs:all>
                <xs:element name="outConfigs" type="configSet" minOccurs="0"/>
                <xs:element name="outUnresolved" type="dnSet" minOccurs="0"/>
            </xs:all>
            <xs:attribute name="cookie" type="xs:string"/>
            <xs:attribute name="response" type="YesOrNo"/>
            <xs:attribute name="errorCode" type="xs:unsignedInt"/>
            <xs:attribute name="errorDescr" type="xs:string"/>
            <xs:attribute name="invocationResult" type="xs:string"/>
        </xs:complexType>
```

Example

Request

```
<configResolveDns
    cookie="<real_cookie>"
    inHierarchical="false">
    <inDns>
        <dn value="sys" />
    </inDns>
</configResolveDns>
```

Response

```
<configResolveDns
    cookie="<real_cookie>"
    commCookie="5/12/0/1009"
    srcExtSys="172.25.103.136"
    destExtSys="172.25.103.136"
    srcSvc="sam_extXMLApi"
    destSvc="resource-mgr_dme"
    response="yes">
    <outConfigs>
        <topSystem
            address="172.25.103.136"
            currentTime="2012-08-01T00:47:44.663"
            dn="sys"
            mode="stand-alone"
            name="localhost"
            systemOrg=""
            systemUpTime="04:04:05:00"/>
    </outConfigs>
    <outUnresolved>
    </outUnresolved>
</configResolveDns>
```

configResolveParent

The following example shows how to retrieve the parent of the managed object for a specified DN.

Request Syntax

```
<xs:element name="configResolveParent" type="configResolveParent"
substitutionGroup="externalMethod"/>
        <xs:complexType name="configResolveParent" mixed="true">
            <xs:attribute name="inHierarchical">
                <xs:simpleType>
                    <xs:union memberTypes="xs:boolean">
                         <xs:simpleType>
                             <xs:restriction base="xs:string">
                                 <xs:enumeration value="no"/>
                                 <xs:enumeration value="yes"/>
                             </xs:restriction>
                         </xs:simpleType>
                    </xs:union>
                </xs:simpleType>
            </xs:attribute>
            <xs:attribute name="cookie" type="xs:string"/>
            <xs:attribute name="response" type="YesOrNo"/>
            <xs:attribute name="dn" type="referenceObject"/>
        </xs:complexType>
```

Response Syntax

```
<xs:element name="configResolveParent" type="configResolveParent"
substitutionGroup="externalMethod"/>
        <xs:complexType name="configResolveParent" mixed="true">
            <xs:all>
                <xs:element name="outConfig" type="configConfig" minOccurs="0"/>
            </xs:all>
            <xs:attribute name="cookie" type="xs:string"/>
            <xs:attribute name="response" type="YesOrNo"/>
            <xs:attribute name="errorCode" type="xs:unsignedInt"/>
            <xs:attribute name="errorDescr" type="xs:string"/>
            <xs:attribute name="invocationResult" type="xs:string"/>
            <xs:attribute name="dn" type="referenceObject"/>
        </xs:complexType>
```

Example

Request

```
<configResolveParent
    cookie="<real_cookie>"
    inHierarchical="false"
    dn="org-root/org-tenant1/org-HR">
</configResolveParent>
```

Response

```
<configResolveParent
    dn="org-root/org-tenant/org-HR"
    cookie="<real_cookie>"
    commCookie="2/12/0/1837"
```

```
    srcExtSys="172.20.101.150"
    destExtSys="172.20.101.150"
    srcSvc="sam_extXMLApi"
    destSvc="service-reg_dme"
    response="yes">
        <outConfig>
        <orgTenant
        descr="tenant123"
        dn="org-root/org-tenant"
        fltAggr="0"
        level="1"
        name="tenant123"/>
    </outConfig>
</configResolveParent>
```

configScope

The following example shows how to return managed objects and details about their configuration.

Request Syntax

```
<xs:element name="configScope" type="configScope" substitutionGroup="externalMethod"/>
        <xs:complexType name="configScope" mixed="true">
            <xs:all>
                <xs:element name="inFilter" type="filterFilter" minOccurs="0"/>
            </xs:all>
            <xs:attribute name="inClass" type="namingClassId"/>
            <xs:attribute name="inHierarchical">
                <xs:simpleType>
                    <xs:union memberTypes="xs:boolean">
                         <xs:simpleType>
                             <xs:restriction base="xs:string">
                                 <xs:enumeration value="no"/>
                                 <xs:enumeration value="yes"/>
                             </xs:restriction>
                         </xs:simpleType>
                    </xs:union>
                </xs:simpleType>
            </xs:attribute>
            <xs:attribute name="inRecursive">
                <xs:simpleType>
                    <xs:union memberTypes="xs:boolean">
                         <xs:simpleType>
                             <xs:restriction base="xs:string">
                                 <xs:enumeration value="no"/>
                                 <xs:enumeration value="yes"/>
                             </xs:restriction>
                         </xs:simpleType>
                    </xs:union>
                </xs:simpleType>
            </xs:attribute>
            <xs:attribute name="cookie" type="xs:string"/>
            <xs:attribute name="response" type="YesOrNo"/>
            <xs:attribute name="dn" type="referenceObject"/>
        </xs:complexType>
```

Response Syntax

```
<xs:element name="configScope" type="configScope" substitutionGroup="externalMethod"/>
        <xs:complexType name="configScope" mixed="true"> <xs:all>
                <xs:element name="outConfigs" type="configSet" minOccurs="0"/>
            </xs:all>
            <xs:attribute name="cookie" type="xs:string"/>
            <xs:attribute name="response" type="YesOrNo"/>
            <xs:attribute name="errorCode" type="xs:unsignedInt"/>
            <xs:attribute name="errorDescr" type="xs:string"/>
            <xs:attribute name="invocationResult" type="xs:string"/>
            <xs:attribute name="dn" type="referenceObject"/>
        </xs:complexType>
```

Example

Request

```
<configScope
    dn="org-root"
    cookie="<real_cookie>"
    inClass="orgOrgRes"
    inHierarchical="false"
    inRecursive="false">
    <inFilter>
    </inFilter>
</configScope>
```

Response

```
<configScope dn="org-root"
    cookie="<real_cookie>"
    commCookie="2/15/0/2a53"
    srcExtSys="10.193.33.120"
    destExtSys="10.193.33.120"
    srcSvc="sam_extXMLApi"
    destSvc="service-reg_dme"
    response="yes">
    <outConfigs>
        <orgOrgCaps
            dn="org-root/org-caps"
            org="512"
            tenant="64"/>
        <orgOrgCounts
            dn="org-root/org-counter"
            org="36"
            tenant="7"/>
    </outConfigs>
</configScope>
```

eventSendHeartbeat

The following example shows how to send an event that indicates the current session is still active.

Request Syntax

```
<xs:element name="eventSendHeartbeat" type="eventSendHeartbeat"
substitutionGroup="externalMethod"/>
        <xs:complexType name="eventSendHeartbeat" mixed="true">
            <xs:attribute name="cookie" type="xs:string"/>
            <xs:attribute name="response" type="YesOrNo"/>
        </xs:complexType>
```

Response Syntax

```
<xs:element name="eventSendHeartbeat" type="eventSendHeartbeat"
substitutionGroup="externalMethod"/>
        <xs:complexType name="eventSendHeartbeat" mixed="true">
            <xs:attribute name="outSystemTime" type="dateTime"/>
            <xs:attribute name="cookie" type="xs:string"/>
            <xs:attribute name="response" type="YesOrNo"/>
            <xs:attribute name="errorCode" type="xs:unsignedInt"/>
            <xs:attribute name="errorDescr" type="xs:string"/>
            <xs:attribute name="invocationResult" type="xs:string"/>
        </xs:complexType>
```

Example

Request

When the client application subscribes to an event or events using eventSubscribeApps or eventSubscribe, the VNMC sends eventSendHeartbeat periodically (default 120 seconds).

Response

```
<eventSendHeartbeat cookie="0/0/0/2a76"
    commCookie=""
    srcExtSys="0.0.0.0"
    destExtSys="0.0.0.0"
    srcSvc=""
    destSvc=""
    response="yes"
    outSystemTime="2010-11-12T20:38:19.630">
</eventSendHeartbeat>
```

eventSubscribe

The following example shows how to send a subscribe request for activity.

Request Syntax

```
<xs:element name="eventSubscribe" type="eventSubscribe"
substitutionGroup="externalMethod"/>
        <xs:complexType name="eventSubscribe" mixed="true">
            <xs:all>
```

```
                <xs:element name="inFilter" type="filterFilter" minOccurs="0"/>
            </xs:all>
            <xs:attribute name="cookie" type="xs:string"/>
            <xs:attribute name="response" type="YesOrNo"/>
        </xs:complexType>
```

Response Syntax

```
<xs:element name="eventSubscribe" type="eventSubscribe"
substitutionGroup="externalMethod"/>
        <xs:complexType name="eventSubscribe" mixed="true">
            <xs:attribute name="cookie" type="xs:string"/>
            <xs:attribute name="response" type="YesOrNo"/>
            <xs:attribute name="errorCode" type="xs:unsignedInt"/>
            <xs:attribute name="errorDescr" type="xs:string"/>
            <xs:attribute name="invocationResult" type="xs:string"/>
        </xs:complexType>
```

Example

Request

```
<eventSubscribe
    cookie="<real_cookie>">
    <inFilter>
    </inFilter>
</eventSubscribe>
```

Response

VNMC sends no response or acknowledgement.

eventSubscribeApps

The following example shows a subscribe request for activity on specified applications. The client application can subscribe to the VNMC system to receive events from different applications. In eventApplication, ip is the IP address for the VM where the application (DME) is running. The client application can subscribe to receive events from the VSG as well, where *ip* should be the IP address for the VSG, and type is managed-endpoint.

Request Syntax

```
<xs:element name="eventSubscribeApps" type="eventSubscribeApps"
substitutionGroup="externalMethod"/>
        <xs:complexType name="eventSubscribeApps" mixed="true">
            <xs:all>
                <xs:element name="inAppList" type="configSet" minOccurs="0"/>
                <xs:element name="inFilter" type="filterFilter" minOccurs="0"/>
            </xs:all>
            <xs:attribute name="cookie" type="xs:string"/>
            <xs:attribute name="response" type="YesOrNo"/>
        </xs:complexType>
```

Response Syntax

```
<xs:element name="eventSubscribeApps" type="eventSubscribeApps"
substitutionGroup="externalMethod"/>
        <xs:complexType name="eventSubscribeApps" mixed="true">
            <xs:attribute name="cookie" type="xs:string"/>
            <xs:attribute name="response" type="YesOrNo"/>
            <xs:attribute name="errorCode" type="xs:unsignedInt"/>
            <xs:attribute name="errorDescr" type="xs:string"/>
            <xs:attribute name="invocationResult" type="xs:string"/>
        </xs:complexType>
```

Example

Request

```
<eventSubscribeApps
    cookie="<real_cookie>"
    commCookie=""
    srcExtSys="0.0.0.0"
    destExtSys="0.0.0.0"
    <inAppList>
        <eventApplication
            ip="10.193.33.101"
            type="service-reg"/>
            <eventApplication
            ip="10.193.33.101"
            type="policy-mgr"/>
        <eventApplication
            ip="10.193.33.101"
            type="mgmt-controller"/>
    </inAppList>
    <inFilter>
    </inFilter>
</eventSubscribeApps>
```

Response

If the request is successful, VNMC sends no response or acknowledgement.

faultAckFault

The following example shows how to send an acknowledgement when a fault is recorded.

Request Syntax

```
<xs:element name="faultAckFault" type="faultAckFault" substitutionGroup="externalMethod"/>
        <xs:complexType name="faultAckFault" mixed="true">
            <xs:attribute name="inId" type="xs:unsignedLong"/>
            <xs:attribute name="cookie" type="xs:string"/>
            <xs:attribute name="response" type="YesOrNo"/>
        </xs:complexType>
```

Response Syntax

```
<xs:element name="faultAckFault" type="faultAckFault" substitutionGroup="externalMethod"/>
        <xs:complexType name="faultAckFault" mixed="true">
            <xs:attribute name="cookie" type="xs:string"/>
            <xs:attribute name="response" type="YesOrNo"/>
            <xs:attribute name="errorCode" type="xs:unsignedInt"/>
            <xs:attribute name="errorDescr" type="xs:string"/>
            <xs:attribute name="invocationResult" type="xs:string"/>
        </xs:complexType>
```

Example

Request

```
<faultAckFault
    inHierarchical="false"
    cookie="<real_cookie>"
    inId="10120" />
```

Response

```
<faultAckFault
    cookie="<real_cookie>"
    commCookie="5/15/0/6c"
    srcExtSys="10.193.33.214"
    destExtSys="10.193.33.214"
    srcSvc="sam_extXMLApi"
    destSvc="resource-mgr_dme"
    response="yes">
</faultAckFault>
```

faultAckFaults

The following example shows how to send an acknowledgement when multiple faults are recorded.

Request Syntax

```
<xs:element name="faultAckFaults" type="faultAckFaults"
substitutionGroup="externalMethod"/>
        <xs:complexType name="faultAckFaults" mixed="true">
            <xs:all>
                <xs:element name="inIds" type="idSet" minOccurs="0"/>
            </xs:all>
            <xs:attribute name="cookie" type="xs:string"/>
            <xs:attribute name="response" type="YesOrNo"/>
        </xs:complexType>
```

Response Syntax

```
<xs:element name="faultAckFaults" type="faultAckFaults"
substitutionGroup="externalMethod"/>
        <xs:complexType name="faultAckFaults" mixed="true">
            <xs:attribute name="cookie" type="xs:string"/>
            <xs:attribute name="response" type="YesOrNo"/>
            <xs:attribute name="errorCode" type="xs:unsignedInt"/>
```

```
            <xs:attribute name="errorDescr" type="xs:string"/>
            <xs:attribute name="invocationResult" type="xs:string"/>
        </xs:complexType>
```

Example

Request

```
<faultAckFaults
    cookie="<real_cookie>">
    <inIds>
        <id value="10656"/>
        <id value="10660"/>
    </inIds>
</faultAckFaults>
```

Response

```
<faultAckFaults
    cookie="<real_cookie>"
    commCookie="11/15/0/505"
    srcExtSys="10.193.34.70"
    destExtSys="10.193.34.70"
    srcSvc="sam_extXMLApi"
    destSvc="mgmt-controller_dme"
    response="yes">
</faultAckFaults>
```

faultResolveFault

The following example shows how to send a response when a fault has been resolved.

Request Syntax

```
<xs:element name="faultResolveFault" type="faultResolveFault"
substitutionGroup="externalMethod"/>
        <xs:complexType name="faultResolveFault" mixed="true">
            <xs:attribute name="inId" type="xs:unsignedLong"/>
            <xs:attribute name="cookie" type="xs:string"/>
            <xs:attribute name="response" type="YesOrNo"/>
        </xs:complexType>
```

Response Syntax

```
<xs:element name="faultResolveFault" type="faultResolveFault"
substitutionGroup="externalMethod"/>
        <xs:complexType name="faultResolveFault" mixed="true">
            <xs:all>
                <xs:element name="outFault" type="configConfig" minOccurs="0"/>
            </xs:all>
            <xs:attribute name="cookie" type="xs:string"/>
            <xs:attribute name="response" type="YesOrNo"/>
            <xs:attribute name="errorCode" type="xs:unsignedInt"/>
            <xs:attribute name="errorDescr" type="xs:string"/>
            <xs:attribute name="invocationResult" type="xs:string"/>
        </xs:complexType>
```

Example

Request

```
<faultResolveFault
    inHierarchical="false"
    cookie="<real_cookie>"
    inId="10120" />
```

Response

```
<faultResolveFault
    cookie="<real_cookie>"
    commCookie="5/15/0/6a"
    srcExtSys="10.193.33.214"
    destExtSys="10.193.33.214"
    srcSvc="sam_extXMLApi"
    destSvc="resource-mgr_dme"
    response="yes">
    <outFault>
        <faultInst
            ack="yes"
            cause="empty-pool"
            changeSet=""
            code="F0135"
            created="2010-11-19T11:02:41.568"
            descr="Virtual Security Gateway pool default is empty"
            dn="org-root/fwpool-default/fault-F0135"
            highestSeverity="minor"
            id="10120"
            lastTransition="2010-11-19T11:02:41.568"
            lc=""
            occur="1"
            origSeverity="minor"
            prevSeverity="minor"
            rule="fw-pool-empty"
            severity="minor"
            tags=""
            type="equipment"/>
    </outFault>
</faultResolveFault>
```

loggingSyncOcns

The following example shows how to retrieve event IDs from DME.

Request Syntax

```
<xs:element name="loggingSyncOcns" type="loggingSyncOcns"
substitutionGroup="externalMethod"/>
        <xs:complexType name="loggingSyncOcns" mixed="true">
            <xs:attribute name="inFromOrZero" type="xs:unsignedLong"/>
            <xs:attribute name="inToOrZero" type="xs:unsignedLong"/>
            <xs:attribute name="cookie" type="xs:string"/>
            <xs:attribute name="response" type="YesOrNo"/>
        </xs:complexType>
```

Response Syntax

```
<xs:element name="loggingSyncOcns" type="loggingSyncOcns"
substitutionGroup="externalMethod"/>
        <xs:complexType name="loggingSyncOcns" mixed="true">
            <xs:all>
                <xs:element name="outStimuli" type="MethodSet" minOccurs="0"/>
            </xs:all>
            <xs:attribute name="cookie" type="xs:string"/>
            <xs:attribute name="response" type="YesOrNo"/>
            <xs:attribute name="errorCode" type="xs:unsignedInt"/>
            <xs:attribute name="errorDescr" type="xs:string"/>
            <xs:attribute name="invocationResult" type="xs:string"/>
        </xs:complexType>
```

Example

Request

```
<loggingSyncOcns
    cookie="<real_cookie>"
    inFromOrZero="0"
    inToOrZero="4567000"/>
```

Response

List of event IDs.

orgResolveElements

Within a specified DN, this example retrieves managed objects that satisfy a query filter and searches managed objects starting at an organization, and optionally in the child organizations.

If there is no organization with that DN, an empty map is returned. If found, it searches managed objects with specified class and filters.

If *inHierarchical* = true, the returned objects will also contain their children. If *inHierarchical* = false, only the matching objects are returned. If *inSingleLevel* = true, only the objects at the starting organization level are returned. If *inSingleLevel* = false, objects in child organizations are also returned.

Request Syntax

```
<xs:element name="orgResolveElements" type="orgResolveElements"
substitutionGroup="externalMethod"/>
        <xs:complexType name="orgResolveElements" mixed="true">
            <xs:all>
                <xs:element name="inFilter" type="filterFilter" minOccurs="0"/>
            </xs:all>
            <xs:attribute name="inClass" type="namingClassId"/>
            <xs:attribute name="inSingleLevel">
                <xs:simpleType>
                    <xs:union memberTypes="xs:boolean">
                        <xs:simpleType>
                            <xs:restriction base="xs:string">
                                <xs:enumeration value="no"/>
                                <xs:enumeration value="yes"/>
                            </xs:restriction>
```

```
                    </xs:simpleType>
                </xs:union>
            </xs:simpleType>
        </xs:attribute>
        <xs:attribute name="inHierarchical">
            <xs:simpleType>
                <xs:union memberTypes="xs:boolean">
                    <xs:simpleType>
                        <xs:restriction base="xs:string">
                            <xs:enumeration value="no"/>
                            <xs:enumeration value="yes"/>
                        </xs:restriction>
                    </xs:simpleType>
                </xs:union>
            </xs:simpleType>
        </xs:attribute>
        <xs:attribute name="cookie" type="xs:string"/>
        <xs:attribute name="response" type="YesOrNo"/>
        <xs:attribute name="dn" type="referenceObject"/>
    </xs:complexType>
```

Response Syntax

```
<xs:element name="orgResolveElements" type="orgResolveElements"
substitutionGroup="externalMethod"/>
    <xs:complexType name="orgResolveElements" mixed="true">
        <xs:all>
            <xs:element name="outConfigs" type="configMap" minOccurs="0">
                <xs:unique name="unique_map_key_5">
                    <xs:selector xpath="pair"/>
                    <xs:field xpath="@key"/>
                </xs:unique>
            </xs:element>
        </xs:all>
        <xs:attribute name="cookie" type="xs:string"/>
        <xs:attribute name="response" type="YesOrNo"/>
        <xs:attribute name="errorCode" type="xs:unsignedInt"/>
        <xs:attribute name="errorDescr" type="xs:string"/>
        <xs:attribute name="invocationResult" type="xs:string"/>
        <xs:attribute name="dn" type="referenceObject"/>
    </xs:complexType>
```

Example

Request

```
<orgResolveElements
    dn="org-root/org-Cola"
    cookie="<real_cookie>"
    commCookie="7/15/0/19"
    inClass="policyPolicySet"
    inSingleLevel="no"
    inHierarchical="no">
    <inFilter>
    </inFilter>
</orgResolveElements>
```

Response

```
<orgResolveElements
    dn="org-root/org-Cola"
```

```
    cookie="<real_cookie>"
    commCookie="7/15/0/19"
    srcExtSys="10.193.33.221"
    destExtSys="10.193.33.221"
    srcSvc="sam_extXMLApi"
    destSvc="policy-mgr_dme"
    response="yes"
    errorCode="0"
    errorDescr="">
    <outConfigs>
        <pair key="pset-default">
            <policyPolicySet
            descr="The default Policy Set"
            dn="org-root/pset-default"
            intId="10082"
            name="default"/>
        </pair>
        <pair key="pset-myPolicySet3">
            <policyPolicySet
            descr=""
            dn="org-root/org-Cola/pset-myPolicySet3"
            intId="12289"
            name="myPolicySet3"/>
        </pair>
        <pair key="pset-policySetSanity">
            <policyPolicySet
            descr=""
            dn="org-root/org-Cola/pset-policySetSanity"
            intId="24627"
            name="policySetSanity"/>
        </pair>
        <pair key="pset-pci_compliance_f">
            <policyPolicySet
            descr=""
            dn="org-root/pset-pci_compliance_f"
            intId="24539"
            name="pci_compliance_f"/>
        </pair>
        <pair key="pset-pci_compliance_h">
            <policyPolicySet
            descr=""
            dn="org-root/pset-pci_compliance_h"
            intId="24541"
            name="pci_compliance_h"/>
        </pair>
    </outConfigs>
</orgResolveElements>
```

orgResolveInScope

The following example shows how the system looks up the organization with the given DN, and (optional) parent organizations recursively to the root. If an organization is not found, an empty map is returned. If found, it searches all pools with specified class and filters.

Note If *inSingleLevel* = false, the system searches parent organizations up to the root directory.

Request Syntax

```
<xs:element name="orgResolveInScope" type="orgResolveInScope"
substitutionGroup="externalMethod"/>
        <xs:complexType name="orgResolveInScope" mixed="true">
            <xs:all>
                <xs:element name="inFilter" type="filterFilter" minOccurs="0"/>
            </xs:all>
            <xs:attribute name="inClass" type="namingClassId"/>
            <xs:attribute name="inSingleLevel">
                <xs:simpleType>
                    <xs:union memberTypes="xs:boolean">
                         <xs:simpleType>
                             <xs:restriction base="xs:string">
                                 <xs:enumeration value="no"/>
                                 <xs:enumeration value="yes"/>
                             </xs:restriction>
                         </xs:simpleType>
                    </xs:union>
                </xs:simpleType>
            </xs:attribute>
            <xs:attribute name="inHierarchical">
                <xs:simpleType>
                    <xs:union memberTypes="xs:boolean">
                         <xs:simpleType>
                             <xs:restriction base="xs:string">
                                 <xs:enumeration value="no"/>
                                 <xs:enumeration value="yes"/>
                             </xs:restriction>
                         </xs:simpleType>
                    </xs:union>
                </xs:simpleType>
            </xs:attribute>
            <xs:attribute name="cookie" type="xs:string"/>
            <xs:attribute name="response" type="YesOrNo"/>
            <xs:attribute name="dn" type="referenceObject"/>
        </xs:complexType>
```

Response Syntax

```
<xs:element name="orgResolveInScope" type="orgResolveInScope"
substitutionGroup="externalMethod"/>
        <xs:complexType name="orgResolveInScope" mixed="true">
            <xs:all>
                <xs:element name="outConfigs" type="configMap" minOccurs="0">
                    <xs:unique name="unique_map_key_6">
                         <xs:selector xpath="pair"/>
                         <xs:field xpath="@key"/>
                    </xs:unique>
                </xs:element>
```

```
            </xs:all>
            <xs:attribute name="cookie" type="xs:string"/>
            <xs:attribute name="response" type="YesOrNo"/>
            <xs:attribute name="errorCode" type="xs:unsignedInt"/>
            <xs:attribute name="errorDescr" type="xs:string"/>
            <xs:attribute name="invocationResult" type="xs:string"/>
            <xs:attribute name="dn" type="referenceObject"/>
        </xs:complexType>
```

Example

Request

```
<orgResolveInScope
    cookie="<real_cookie>"
    dn="org-root/org-Cola"
    inClass="policyVirtualNetworkServiceProfile"
    inHierarchical="true"
    InSingleLevel="false" >
    <inFilter>
        <eq class="policyVirtualNetworkServiceProfile"
        property="name"
        value="spsanity" />
    </inFilter>
</orgResolveInScope>
```

Response

```
<orgResolveInScope
    dn="org-root/org-Cola"
    cookie="<real_cookie>"
    commCookie="7/15/0/1c35"
    srcExtSys="10.193.34.70"
    destExtSys="10.193.34.70"
    srcSvc="sam_extXMLApi"
    destSvc="policy-mgr_dme"
    response="yes">
    <outConfigs>
        <pair key="vnsp-spsanity">
            <policyVirtualNetworkServiceProfile
                childAction="deleteNonPresent"
                descr=""
                dn="org-root/org-Cola/vnsp-spsanity"
                intId="82018"
                name="spsanity"
                policySetNameRef="policySetSanity"
                vnspId="41">
            <policyVnspAVPair
                childAction="deleteNonPresent"
                descr=""
                id="1"
                intId="82019"
                name=""
                rn="vnsp-avp-1">
                <policyAttributeValue
                    childAction="deleteNonPresent"
                    id="1"
                    rn="attr-val1"
                    value="DEV"/>
                <policyAttributeDesignator
                    attrName="dept"
                    childAction="deleteNonPresent"
```

```
                    rn="attr-ref"/>
                </policyVnspAVPair>
            </policyVirtualNetworkServiceProfile>
        </pair>
    </outConfigs>
</orgResolveInScope>
```

poolResolveInScope

The following example shows how the system looks up the pool with the given DN, and (optional) parent pools recursively to the root. If no pool exists, an empty map is returned. If found, the system searches all pools with the specified class and filters.

Note If *inSingleLevel* = false, the system searches parent pools up to the root directory.

Request Syntax

```
<xs:element name="poolResolveInScope" type="poolResolveInScope"
substitutionGroup="externalMethod"/>
        <xs:complexType name="poolResolveInScope" mixed="true">
            <xs:all>
                <xs:element name="inFilter" type="filterFilter" minOccurs="0"/>
            </xs:all>
            <xs:attribute name="inClass" type="namingClassId"/>
            <xs:attribute name="inSingleLevel">
                <xs:simpleType>
                    <xs:union memberTypes="xs:boolean">
                        <xs:simpleType>
                            <xs:restriction base="xs:string">
                                <xs:enumeration value="no"/>
                                <xs:enumeration value="yes"/>
                            </xs:restriction>
                        </xs:simpleType>
                    </xs:union>
                </xs:simpleType>
            </xs:attribute>
            <xs:attribute name="inHierarchical">
                <xs:simpleType>
                    <xs:union memberTypes="xs:boolean">
                        <xs:simpleType>
                            <xs:restriction base="xs:string">
                                <xs:enumeration value="no"/>
                                <xs:enumeration value="yes"/>
                            </xs:restriction>
                        </xs:simpleType>
                    </xs:union>
                </xs:simpleType>
            </xs:attribute>
            <xs:attribute name="cookie" type="xs:string"/>
            <xs:attribute name="response" type="YesOrNo"/>
            <xs:attribute name="dn" type="referenceObject"/>
        </xs:complexType>
```

Response Syntax

```
<xs:element name="poolResolveInScope" type="poolResolveInScope"
substitutionGroup="externalMethod"/>
        <xs:complexType name="poolResolveInScope" mixed="true">
            <xs:all>
                <xs:element name="outConfigs" type="configMap" minOccurs="0">
                    <xs:unique name="unique_map_key_9">
                         <xs:selector xpath="pair"/>
                         <xs:field xpath="@key"/>
                    </xs:unique>
                </xs:element>
            </xs:all>
            <xs:attribute name="cookie" type="xs:string"/>
            <xs:attribute name="response" type="YesOrNo"/>
            <xs:attribute name="errorCode" type="xs:unsignedInt"/>
            <xs:attribute name="errorDescr" type="xs:string"/>
            <xs:attribute name="invocationResult" type="xs:string"/>
            <xs:attribute name="dn" type="referenceObject"/>
        </xs:complexType>
```

Example

Request

```
<poolResolveInScope
dn="org-root/org-tenant1"
cookie="<real_cookie>"
/>
```

Response

```
<poolResolveInScope
    dn="org-root/org-cisco"
    cookie="<real_cookie>"
    commCookie="5/12/0/19"
    srcExtSys="172.25.103.136"
    destExtSys="172.25.103.136"
    srcSvc="sam_extXMLApi"
    destSvc="resource-mgr_dme"
    response="yes">
    <outConfigs>
        <pair key="fwpool-default">
            <fwPool
                assigned="0"
                descr="Default Pool of firewall resources"
                dn="org-root/fwpool-default"
                fltAggr="65536"
                id="1"
                intId="10066"
                name="default"
                size="0"/>
        </pair>
    </outConfigs>
</poolResolveInScope>
```