Cisco IMC XML Schema Definition Files

This chapter includes the following sections:

About the Cisco IMC XML Schema Definition Files

The C-Series XML API provides users with input XML Schema Definition (xsd) files for every model and a schema definition file for the output:

- RACK-IN.xsd — This document defines the XML document structure for a valid XML request that the Cisco IMC XML API accepts. It also specifies the classes and attributes that you can provide to the XML API configConfMo (Set) requests.
- RACK-OUT.xsd — This document defines the XML document structure for a valid XML response that the Cisco IMC XML API displays. It also specifies the classes and attributes that must appear in the XML API responses.

You can obtain these files from the Cisco IMC at:

RACK-IN.xsd — **https://<Cisco IMC-IP>/visore/RACK-IN.xsd**

RACK-OUT — **https://<Cisco IMC-IP>/visore/RACK-OUT.xsd**

You also can download these files from the Cisco Developer Network, at: http://developer.cisco.com/web/unifiedcomputing/c-series-cimc-xml-api.

You can use one of the available open source tools to validate the XML document against the XML schema files. In the examples used in this section, xmllint available for download at: www.xmlsoft.org is used as the tool for validation. You also can use the xml schema validation feature of a programming language, for example xerces in Java, for this validation.

Examples of RACK-IN.xsd Usage

Validating an XML Request Using RACK-IN.xsd

Example of Use of the RACK-IN.xsd for an Invalid configResolveClass Request

Request:

```
$cat myXMLRequest.xml
<configResolveClass cookie="1360626069/7189c2b0-d57b-157b-8002-f4759de53d50"
inHierarchical="false"/>
```

Validating the request:

```
/usr/bin/xmllint –schema  ./RACK-IN.xsd  myXMLRequest.xml
<configResolveClass cookie="1360626069/7189c2b0-d57b-157b-8002-f4759de53d50"
inHierarchical="false"/>
```

Response:

```
myXMLRequest.xml:1: element configResolveClass:
Schemas validity error : Element 'configResolveClass': The attribute 'classId' is required
 but missing.
```

In the preceding example, validation of the XML request fails and displays an error because the 'classId' is missing in the request.

Example of Use of the RACK-IN.xsd for a Valid configResolveClass Request

Request:

```
$cat myXMLRequest.xml
<configResolveClass cookie="1360626069/7189c2b0-d57b-157b-8002-f4759de53d50"
inHierarchical="false" classId="topSystem"/>
```

Request:

```
/usr/bin/xmllint -schema ./RACK-IN.xsd  myXMLRequest.xml
<configResolveClass cookie="1360626069/7189c2b0-d57b-157b-8002-f4759de53d50"
inHierarchical="false" classId="topSystem"/>
```

In the preceding example, validation of the XML request for a classResolveClass is successful and the response is displayed.

Example of Use of the RACK-IN.xsd for an Invalid configConfMo Request

Request:

```
$cat setRackUnit.xml
<configConfMo cookie="1300242644/ad04d239-d1aa-498d-b074-ccb923066003"
dn="sys/rack-unit-1" inHierarchical="false">
  <inConfig>
     <computeRackUnit adminPower="down" usrLbl="UCS server For Demo"
      availableMemory="16384" dn="sys/rack-unit-1"/>
  </inConfig>
</configConfMo>
```

Validating the request:

```
/usr/bin/xmllint -schema ./RACK-IN.xsd  /setRackUnit.xml
<configConfMo  cookie="1300242644/ad04d239-d1aa-498d-b074-ccb923066003"
dn="sys/rack-unit-1"inHierarchical="false">
  <inConfig>
    <computeRackUnit adminPower="down" usrLbl="UCS server For Demo"
     availableMemory="16384" dn="sys/rack-unit-1"/>
  </inConfig>
</configConfMo>
```

Response:

```
/setRackUnit.xml:3: element computeRackUnit: Schemas validity error :
Element 'computeRackUnit', attribute 'availableMemory': The attribute 'availableMemory' is
 not allowed.
/setRackUnit.xml fails to validate
```

The availableMemory attribute is read-only in the computeRackUnitclass. You can view the read/write attributes in computeRackUnit that can be set using configCongMo XML by looking at the computeRackUnit definition in RACK-IN.xsd. A sample snippet is as follows:

```
<!--computeRackUnit-->
        <xs:element name="computeRackUnit" type="computeRackUnit"
substitutionGroup="managedObject"/>
        <xs:complexType name="computeRackUnit" mixed="true">

            <xs:attribute name="adminPower">
                <xs:simpleType>
                    <xs:restriction base="xs:string">
                         <xs:enumeration value="up"/>
                         <xs:enumeration value="down"/>
                         <xs:enumeration value="soft-shut-down"/>
                         <xs:enumeration value="cycle-immediate"/>
                         <xs:enumeration value="hard-reset-immediate"/>
                         <xs:enumeration value="bmc-reset-immediate"/>
                         <xs:enumeration value="bmc-reset-default"/>
                         <xs:enumeration value="cmos-reset-immediate"/>
                         <xs:enumeration value="diagnostic-interrupt"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:attribute>

            <xs:attribute name="usrLbl">
                <xs:simpleType>
                    <xs:restriction base="xs:string">
                         <xs:pattern value="[
!#$%&amp;\(\)\*\+,\-\./:;\?@\[\]_\{\|\}~a-zA-Z0-9]{0,64}"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:attribute>

            <xs:attribute name="dn" type="referenceObject"/>
            <xs:attribute name="rn" type="referenceRn"/>
            <xs:attribute name="status" type="objectStatus"/>
        </xs:complexType>
```

Example of Use of the RACK-IN.xsd for an Invalid configConfMo Request

Request:

```
$ cat setBootOrder.xml
<configConfMo cookie="1360205300/79c672f0-d519-1519-8004-30339ee53d50"
inHierarchical="true" dn="sys/rack-unit-1/boot-policy">
  <inConfig>
    <lsbootDef dn="sys/rack-unit-1/boot-policy"   name="boot-policy"  purpose="operational"
 rebootOnUpdate="no">
      <lsbootVirtualMedia access="read-only" order="2" type="virtual-media"
rn="vm-read-only"/>
      <lsbootVirtualMedia access="read-write" order="3" type="virtual-media"
rn="vm-read-write"/>
      <lsbootLan rn="lan-read-only" access="read-only" order="4" prot="pxe" type="lan"/>
      <lsbootStorage rn="storage-read-write" access="read-write" order="1" type="storage">
       <lsbootLocalStorage rn="local-storage"/>
      </lsbootStorage>
      <lsbootEfi rn="efi-read-only" access="read-only" order="5" type="efi"/>
    </lsbootDef>
  </inConfig>
</configConfMo>
```

Validating the request:

```
/usr/bin/xmllint -schema ./RACK-IN.xsd ./setBootOrder.xml
<configConfMo cookie="1360205300/79c672f0-d519-1519-8004-30339ee53d50"
inHierarchical="true" dn="sys/rack-unit-1/boot-policy">
  <inConfig>
    <lsbootDef dn="sys/rack-unit-1/boot-policy" name="boot-policy" purpose="operational"
rebootOnUpdate="no">
      <lsbootVirtualMedia access="read-only" order="2" type="virtual-media"
rn="vm-read-only"/>
      <lsbootVirtualMedia access="read-write" order="3" type="virtual-media"
rn="vm-read-write"/>
      <lsbootLan rn="lan-read-only" access="read-only" order="4" prot="pxe" type="lan"/>
      <lsbootStorage rn="storage-read-write" access="read-write" order="1" type="storage">
        <lsbootLocalStorage rn="local-storage"/>
      </lsbootStorage>
      <lsbootEfi rn="efi-read-only" access="read-only" order="5" type="efi"/>
    </lsbootDef>
  </inConfig>
</configConfMo>
```

Response:

```
./setBootOrder.xml:3: element lsbootDef: Schemas validity error :
Element 'lsbootDef', attribute 'name': The attribute 'name' is not allowed.
./setBootOrder.xml:3: element lsbootDef: Schemas validity error :
Element 'lsbootDef', attribute 'purpose': The attribute 'purpose' is not allowed.
./setBootOrder.xml fails to validate
```

Name and purpose attributes of class lsBootDef are read-only and cannot be used in the configConMo/set request.

Example of Use of the RACK-IN.xsd for a Valid configConfMo Request

Request:

```
usr/bin/xmllint -schema ./RACK-IN.xsd  /setRackUnit.xml
<configConfMo  cookie="1300242644/ad04d239-d1aa-498d-b074-ccb923066003"
 dn="sys/rack-unit-1" inHierarchical="false">
  <inConfig>
    <computeRackUnit adminPower="down" usrLbl="UCS server For Demo" dn="sys/rack-unit-1"/>
</inConfig>
</configConfMo>
```

xsd file validates the request and completes the configuration.

Examples of RACK-OUT.xsd Usage

Validating an XML Request Using RACK_OUT.xsd

Example of Use of the RACK-OUT.xsd for a configResolveClass Request

Request:

```
<configResolveClass cookie="1360632361/e892fa10-d57c-157c-8003-f4759de53d50"
inHierarchical="false" classId="computeRackUnit"/>'
https://172.xx.219.xx/nuova | xmllint -format -  |  xmllint -schema ./RACK-OUT.xsd -
```

Response:

```
<configResolveClass cookie="1360632361/e892fa10-d57c-157c-8003-f4759de53d50"
response="yes" classId="computeRackUnit">
  <outConfigs>
    <computeRackUnit dn="sys/rack-unit-1" adminPower="policy" availableMemory="16384"
model="UCSC-server-name"
     memorySpeed="1333" name="UCS server_name" numOfAdaptors="1" numOfCores="16"
numOfCoresEnabled="16" numOfCpus="2"
     numOfEthHostIfs="2" numOfFcHostIfs="3" numOfThreads="32" operPower="on"
```

```
     originalUuid="2E5D2295-F32D-48C9-BE8E-BAD36BE174FB" presence="equipped" serverId="1"
serial="FCH1551V030"
    totalMemory="16384" usrLbl="SL2_=+@#$-;,./\" uuid="2E5D2295-F32D-48C9-BE8E-BAD36BE174FB"

     vendor="Cisco Systems Inc"/>
  </outConfigs>
</configResolveClass>
```

Rack-out.xsd validates the output successfully.

Example of Use of the RACK-OUT.xsd for a configResolveClass Request

Request:

```
$ /usr/bin/curl -k -d'<configResolveClass
cookie="1361150931/a5bacff0-d5f5-15f5-8007-f4759de53d50"
inHierarchical="true" classId="topSystem"/>'  https://172.29.219.74/nuova >
Cxxx_complete_MIT.xml
real    0m35.065s
user    0m0.016s
sys     0m0.044s
```

Validating the response:

```
$ ls -l Cxxx_complete_MIT.xml
-rw-r--r--. 1 sajaffer eng58 64905 Feb 17 17:46 Cxxx_complete_MIT.xml

$ /usr/bin/xmllint -schema   RACK-OUT.xsd   Cxxx_complete_MIT.xml 1>/dev/null
Cxxx_complete_MIT.xml validates
```

The preceding XML requests retrieve the complete management information tree of a C-Series server and validate the response against RACK-OUT.xsd