Document Number	EDCS-
Created By	Ding Hua

CCSP SNMP Protocol Agent Design Specification

Reviewers

Department	Name/Title
Development Engineering	Ruiqi Tian, Kamel Chbaro
DevTest Engineering	

The departments and/or individuals listed above should be notified in advance and given a sufficient time period to review this document. The Project Team determines requirements for approval according to the scope of the project.

Modification History

Revision	Date	Originator	Comments
1.0	07/07/2014	Ding Hua	Initial draft

Table of Contents

1 Audience

The audience for this document includes:

- Developers who will develop MIB objects within CCSP architecture
- Code review teams who use this document as a guide to evaluate implementation completeness
- Product Instantiation Teams that use this component in point products
- Test Teams will use this document to build test plans and harnesses to exercise the component interfaces

2 Component Overview

SNMP Protocol Agent (PA) provides the solution for SNMP Protocol to access CCSP Data Model.

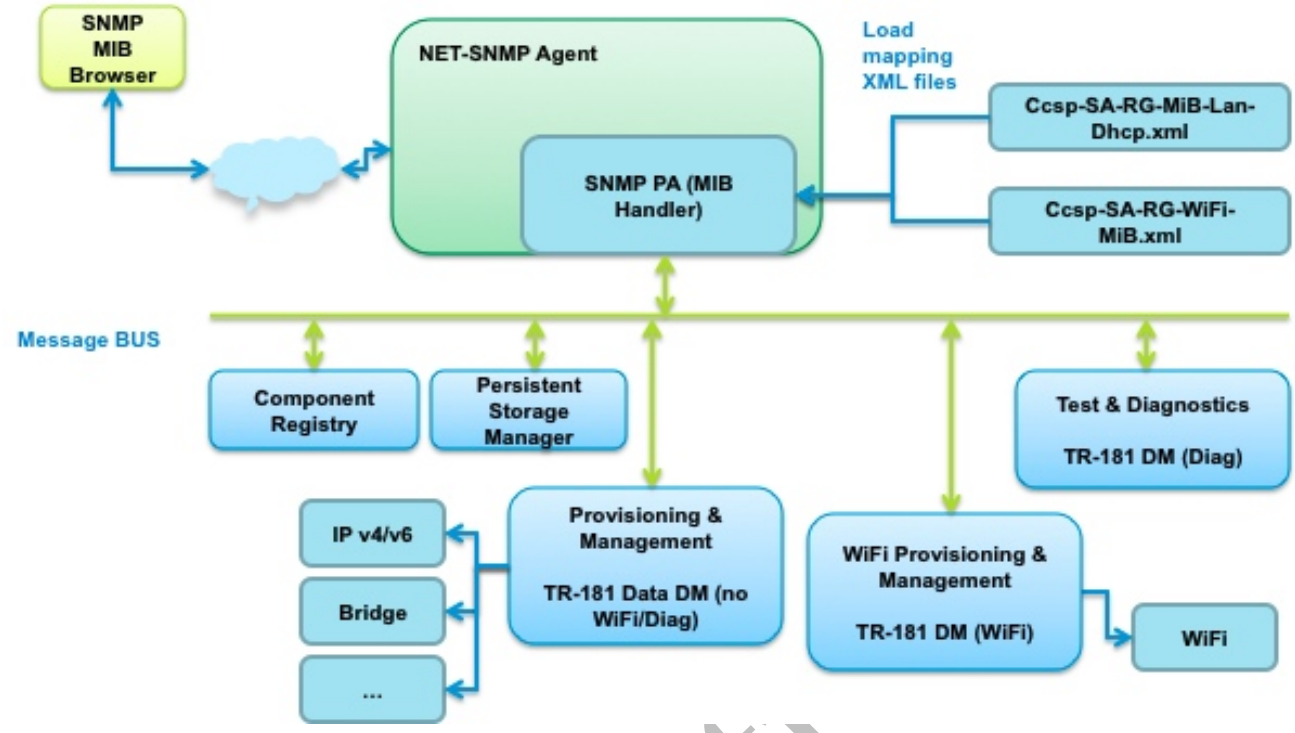
Externally, SNMP protocol agent works like a regular SNMP agent. It takes coming SNMP requests and sends back the corresponding results or errors.

Internally, SNMP protocol agent processes the SNMP requests and translates those into CCSP Messages. Those messages go on to CCSP Message Bus and come back with response messages. SNMP protocol agent processes the response messages and response to SNMP requests accordingly.

In order to translate between SNMP requests and CCSP Messages, SNMP protocol agent loads the XML files describing how MIB objects are mapped to Data Model objects.

3 Component Architecture

This is the architecture diagram of SNMP Protocol Agent



CCSP SNMP Protocol Agent is implemented as a NET-SNMP MIB handler. During startup, it loads all the XML files defining the following:

- MIB object – Data Model object mapping
- Custom callback APIs

After it is up, CCSP SNMP Protocol Agent processes the SNMP requests handed to it by NET-SNMP agent, send the translated CCSP Messages to the destination components, processes the responses and provide the result back to SNMP agent.

4 Design Goals and Considerations

1. Share the same CCSP Data Model implementation used for TR-069, Web GUI, CLI. Eliminate the need to redevelop the MIBs for what CCSP Data Model has supported, but to develop the MIB to Data Model mapping XML file.
2. Open source NET-SNMP agent is utilized as the base for CCSP Snmp Protocol Agent.
3. The MIB to Data Model mapping file should be human readable and easy to edit. It could be one or multiple files.
4. There are scalar and table mibs in SNMP while objects and tables in TR data model. Ideally scalar mibs will be mapped to an Data Model object and table mibs will be mapped to a Data Model table object. It should be considered special cases such as a scalar mib mapping to a specific entry in a data model table.

5. For the future MIB support, only the mapping file should be updated to add the new mapping and/or needed custom callbacks. The handler should remain intact.
6. The data type definition is different between MIBs and CCSP Data Model. The data type mapping definitions are expected in the mapping file.
7. The index of MIB table may be different from the instance numbers in TR data model. The mapping definitions are expected too.

5 MIB to Data Model Mapping XML Schema

5.1 Root Node

The root node is named as “mib2DM”. It is constructed with a set of scalar mib groups, mib tables and some informational nodes.

Below is a piece of schema of the node and an example.

```
<xs:element name="mib2DM">
 <xs:complexType>
  <xs:sequence>
   <xs:element name="version" type="xs:positiveInteger"/>
   <xs:element name="name" type="xs:string"/>
   <xs:element name="lastUpdate" type="xs:string"/>
   <xs:element name="author" type="xs:string"/>
   <!-- Add the library name if callbacks are needed. -->
   <xs:element name="library" type="xs:string" minOccurs=”0”/>
   <xs:element name="scalarGroups" type="scalarGroupArray" minOccurs="0" />
   <xs:element name="mibTables" type="mibTableArray" minOccurs="0" />
  </xs:sequence>
 </xs:complexType>
</xs:element>

Example file:
<?xml version="1.0" encoding="utf-8" ?>
<mib2DM>
    <version>1</version>
   <name>clabWifiMib</name>
    <lastUpdate>04/17/2012 </lastUpdate>
    <author>CCSP MIB Team </author>
    <scalarGroups>
       <scalarGroup>…</scalerGroup>
       <scalarGroup>…</scalerGroup>
   </scalarGroups>
   <mibTables>
     <mibTable>…</mibTable>
     <mibTable>…</mibTable>
     <mibTable>…</mibTable>
  </mibTables>
</mib2DM>
```

5.2 MIB Custom Callback APIs

If special logic is required for certain MIB objects beyond MIB – Data Model mapping can support, custom callback functions can be utilized.

The callback prototype is:

```
typedef int
 (*handleRequestsProc)
```

```
(
        netsnmp_mib_handler             *handler,
        netsnmp_handler_registration    *reginfo,
        netsnmp_agent_request_info      *reqinfo,
        netsnmp_request_info            *requests
);
```

This callback will be called before SNMP Protocol Agent MIB handler processing the requests. If the callback takes care of certain request and it doesn't want to be processed anymore, the "requests->processed" should be set to 1. In this case, the MIB handler will ignore this request and move on.

The XML schema for custom callbacks is:

```
<!-- Definition of callbackInfo -->
<xs:complexType name="callbackInfo">
  <xs:sequence>
   <!--  The prototype of the "handleRequests":
    typedef int
    (*handleRequestsProc)
          (
             netsnmp_mib_handler            *handler,
             netsnmp_handler_registration   *reginfo,
             netsnmp_agent_request_info     *reqinfo,
             netsnmp_request_info           *requests
          );
    This callback will be called before Ccsp generic MIB handler.
    Whenever a request was handled in this callback, you may set "request->processed" to 1.  -->
   <xs:element name="handleRequests" type="xs:string" minOccurs="0" />
   <!--  The prototype of the "refreshCache":
    typedef int
    (*refreshCacheProc)
          (
             netsnmp_tdata*                 table
          );
    This callback will be called first in RefreshCache of Ccsp generic MIB handler.
   -->
   <xs:element name="refreshCache" type="xs:string" minOccurs="0" />
  </xs:sequence>
 </xs:complexType>
```

5.3 Node of "mapping"

Mapping is consisted of two parts,

- MIB object, identified by the OID
- Data Model object/parameter, identified by the Data Model object/parameter name.

The scema is as follows:

```
<!-- Definition of mibInfo -->
<xs:complexType name="mibInfo">
  <xs:sequence>
   <xs:element name="lastOid" type="xs:positiveInteger" />
   <!— The “name” will not be used by SNMP agent, but it will be helpful when editing the mapping file. It’s optional. -->
   <xs:element name="name" type="xs:string" minOccurs=”0” />
   <xs:element name="access" type="mibAccess" />
   <!— The data type of this mib -->
   <xs:element name="dataType" type="xs:string" />
   <xs:element name="range" type="rangeInfo" minOccurs=”0”/>
  </xs:sequence>
 </xs:complexType>


<!-- Definition of dmInfo -->
<xs:complexType name="dmInfo">
  <xs:sequence>
   <!—It defines the mapped Data Model parameter name. It could be under an object or a table.  -->
   <!— For instance, “Device.DeviceInfo.SoftwareVersion” or “Device.Hosts.Host.%d.Name”. -->
   <xs:element name="paramName" type="xs:string" />
   <xs:element name="dataType" type="trDataType " />
   <!— Enumeration is an integer in MIB while a string value in TR data model. We put one single string including the whole mapping. -- >
   <!— For instance “None(0), Requested(1), Complete(2), Error(3)”. The maps are separated by a ‘,’.-->
   <!--  The bitmask type is also supported. It’s a choice of data type extension here -->
     <xs:choice minOccurs=”0”>
       <xs:element name="enumeration" type="xs:string" />
      <xs:element name="bitmask" type="xs:string" />
   </xs:choice>
  </xs:sequence>
 </xs:complexType>

<!-- Definition of mappingInfo -->
<xs:complexType name="mappingInfo">
  <xs:sequence>
   <xs:element name="mib" type="mibInfo" />
   <!—Certain mibs may not need a mapping in DM, such as “StorageType”. Hence it’s optional. -->
   <xs:element name="dm" type="dmInfo" minOccurs=”0” />
  </xs:sequence>
 </xs:complexType>
```

5.4 Node of "scalarGroup"

This node defines how scalar MIB objects are mapped to Data Model objects:

```
<!—Definition of a scalarGroupInfo -->
<xs:complexType name="scalarGroupInfo">
  <xs:sequence>
   <!-- Unique name of group name. It could be the name of baseOid. It will be used for the hook/callback registration. -->
   <xs:element name="name" type="xs:string"/>
   <!-- "baseOid" is the base OID string for this group of scalar mibs. All the mibs have to be in the same OID layer. -->
   <!-- Otherwise, put them in another scalarGroup. -->
   <xs:element name="baseOid" type="xs:string"/>
   <!-- "enabled" is defined in case this group is not ready at back-end yet or deprecated -- >
   <!-- If it's not enabled, the MIB handler will not load it. It's enabled if the node doesn't present (by default) -->
   <xs:element name="enabled" type="xs:boolean" minOccurs="0"/>
   <!-- Cache is always recommended for performance concerns. The default is 30 seconds -->
   <xs:element name="cacheTimeout" type="xs:nonNegativeInteger" minOccurs="0" />
   <!-- In some special case, the scalar mibs may map to an entry in Data Model Table. -->
   <!-- This optional node provides conditional mapping. For instance, "Wifi.Radio.%d.Frequency=2.4" -->
   <xs:element name="mapToEntry" type="xs:string" minOccurs="0" />
   <!-- If callback/hook functions are needed for this group, add it here. -->
   <xs:element name="callbacks" type="callbackInfo " minOccurs="0" />
    <!-- a set of mibs -->
    <xs:element name="mapping" type="mappingInfo" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
 </xs:complexType>
```

5.5 Node of "mibTable"

Because table objects covers more than parameters, more work and functionalities are included:

- Index mapping – for MIB and Data Model table objects not sharing the same index scheme
- Filtering – Data Model table may contains more instances than SNMP wants. Filtering criteria can be given to pick only the entries matching the criteria. The criteria can be on parameter level, e.g., "Device.Routing.Router.1.IPv4Forwarding.%d.StaticRoute = true".

```
<!-- Definition of indexInfo -->
<xs:complexType name="indexInfo">
  <xs:sequence>
   <xs:element name="mib" type="mibInfo" />
    <xs:choice>
      <!— The mib may be mapped to the instance number of data model entry. It's false by default. -->
      <xs:element name="mapToInsNumber" type="type:mapToInsInfo" />
```

```
      <!— If it's not mapped to instance number, it will be mapped to a data model parameter. -->
      <xs:element name="dm" type="dmInfo" />
    </xs:choice>
  </xs:sequence>
 </xs:complexType>

<!-- Definition of a mibTableInfo -->
<xs:complexType name="mibTableInfo">
  <xs:sequence>
    <!-- Mib table name. It will be used for the hook/callback registration. -->
    <xs:element name="name" type="xs:string"/>
    <!-- The fully OID of the mib table (not entry). For instance "1,3,6,1,4,8072,2,2,2".-->
    <!-- Otherwise, put them in another scalarGroup. -->
    <xs:element name="tableOid" type="xs:string"/>
    <!-- "enabled" is defined in case this group is not ready at back-end yet or deprecated -- >
    <!-- If it's not enabled, the MIB handler will not load it. It's enabled if the node doesn't present (by default) -->
    <xs:element name="enabled" type="xs:boolean" minOccurs="0"/>
    <!-- Whether the table is writable or not. It's not by default. -->
    <xs:element name="writable" type="xs:boolean" minOccurs="0"/>
    <!-- Specify the maxi entries in the table. Default is 16. -->
    <xs:element name="maxEntries" type="xs:positiveInteger" minOccurs="0"/>
    <!-- Cache is always recommended for performance concerns. The default is 30 seconds -->
    <xs:element name="cacheTimeout" type="xs:nonNegativeInteger" minOccurs="0" />
    <!-- In some special case, the mib table maps to some entries in a DM table. -->
    <!-- The filter may be added here to achieve this goal. For instance, "Routing.%d.type = static" -->
    <xs:element name="mapToEntries" type="xs:string" minOccurs="0" />
    <!-- If callback/hook functions are needed for this group, add it here. -->
    <xs:element name="callbacks" type="callbackInfo " minOccurs="0" />
     <!-- a set of indexs -->
     <xs:element name="index" type="indexInfo" minOccurs="1" maxOccurs="8"/>
     <!-- a set of mibs -->
     <xs:element name="mapping" type="mappingInfo" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
 </xs:complexType>
```

5.6 Complete Schema

```
<?xml version="1.0"encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

<!-- Definition of mibAccess -->
<xs:simpleType name="mibAccess">
  <xs:restriction base="xs:string">
      <xs:enumeration value="ReadOnly" />
```

```
        <xs:enumeration value="ReadWrite" />
       <!—Certain mibs may be an action or require empty in value such as "password". We treat them as "WriteOnly". -->
        <xs:enumeration value="WriteOnly" />
       <!— For MIB as an index, it is not accessible. -->
        <xs:enumeration value="NoAccess" />
</xs:restriction>
 </xs:simpleType>

<!-- Definition of callbackInfo -->
<xs:complexType name="callbackInfo">
  <xs:sequence>
    <!--  The prototype of the "handleRequests":
      typedef int
      (*handleRequestsProc)
            (
               netsnmp_mib_handler              *handler,
               netsnmp_handler_registration   *reginfo,
               netsnmp_agent_request_info   *reqinfo,
               netsnmp_request_info             *requests
            );
      This callback will be called before Ccsp generic MIB handler.
      Whenever a request was handled in this callback, you may set "request->processed" to 1.  -->
    <xs:element name="handleRequests" type="xs:string" minOccurs="0" />
    <!--  The prototype of the "refreshCache":
      typedef int
      (*refreshCacheProc)
            (
               netsnmp_tdata*                    table
            );
      This callback will be called first in RefreshCache of Ccsp generic MIB handler.
    -->
    <xs:element name="refreshCache" type="xs:string" minOccurs="0" />
  </xs:sequence>
 </xs:complexType>

<!-- Definition of rangeInfo -->
<xs:complexType name="rangeInfo">
  <xs:sequence>
    <!— This is defined for the limitation. It's a value limitation for an "integer" and size limitation for a "string". -->
    <xs:element name="min" type="xs:integer" minOccurs="0" />
    <xs:element name="max" type="xs:integer" minOccurs="0" />
  </xs:sequence>
 </xs:complexType>

<!-- Definition of mibInfo -->
<xs:complexType name="mibInfo">
  <xs:sequence>
    <xs:element name="lastOid" type="xs:positiveInteger" />
    <!— The "name" will not be used by SNMP agent, but it will be helpful when editing the mapping file. It's optional. -->
    <xs:element name="name" type="xs:string" minOccurs="0" />
```

```
    <xs:element name="access" type="mibAccess" />
    <!— The data type of this mib -->
    <xs:element name="dataType" type="xs:string" />
    <xs:element name="range" type="rangeInfo" minOccurs=”0”/>
   </xs:sequence>
  </xs:complexType>

<!-- Definition of trDataType -->
<xs:simpleType name=" trDataType ">
   <xs:restriction base=”xs:string”>
        <xs:enumeration value=”string” />
        <xs:enumeration value=”int” />
        <xs:enumeration value=”unsignedInt” />
        <xs:enumeration value=”boolean” />
        <xs:enumeration value=”dateTime” />
        <xs:enumeration value=”base64” />
</xs:restriction>
  </xs:simpleType>

<!-- Definition of dmInfo -->
<xs:complexType name="dmInfo">
   <xs:sequence>
    <!—It defines the mapped Data Model parameter name. It could be under an object or a table.  -->
    <!— For instance, “Device.DeviceInfo.SoftwareVersion” or “Device.Hosts.Host.%d.Name”. -->
    <xs:element name="paramName" type="xs:string" />
    <xs:element name="dataType" type="trDataType " />
    <!— Enumeration is an integer in MIB while a string value in TR data model. We put one single string including the whole mapping. -- >
    <!— For instance “None(0), Requested(1), Complete(2), Error(3)”. The maps are separated by a ‘,’.-->
   <!--  The bitmask type is also supported. It’s a choice of data type extension here -->
     <xs:choice minOccurs=”0”>
       <xs:element name="enumeration" type="xs:string" />
      <xs:element name="bitmask" type="xs:string" />
    </xs:choice>
  </xs:sequence>
 </xs:complexType>

<!-- Definition of mappingInfo -->
<xs:complexType name="mappingInfo">
   <xs:sequence>
    <xs:element name="mib" type="mibInfo" />
    <!—Certain mibs may not need a mapping in DM, such as “StorageType”. Hence it’s optional. -->
    <xs:element name="dm" type="dmInfo" minOccurs=”0” />
   </xs:sequence>
 </xs:complexType>
```

```
<!-- Definition of mapToInsInfo -->
<xs:complexType name="mapValueInfo">
  <xs:sequence>
   <!—Sometimes that index value may map to a different instance number value -->
   <!-- For instance, ifIndex=32 map to ins “1”, ifIndex=112 map to “2” -->
   <!-- Hence it will be <from>32</from><to>1</to>… -->
   <xs:element name="from" type="xs:positiveInteger"/>
   <xs:element name="to" type="xs:positiveInteger"/>
  </xs:sequence>
 </xs:complexType>

<!-- Definition of mapToInsInfo -->
<xs:complexType name="mapToInsInfo">
  <xs:sequence>
   <!-- specify the table object whose instance number the index will map to  -->
   <!-- It could be multiple layer tables if multiple indexes exist. -->
   <!-- For instance: “Device.abcTable.%d.childTable.” -->
   <xs:element name="tableObj" type="xs:string" />
   <!-- If there’s no map defined, the index value will equal to insNumber. -->
   <xs:element name="map" type="mapValueInfo" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
 </xs:complexType>

<!-- Definition of indexInfo -->
<xs:complexType name="indexInfo">
  <xs:sequence>
   <xs:element name="mib" type="mibInfo" />
    <xs:choice>
      <!— The mib may be mapped to the instance number of data model entry. It’s false by default. -->
      <xs:element name="mapToInsNumber" type="type:mapToInsInfo" />
      <!— If it’s not mapped to instance number, it will be mapped to a data model parameter. -->
      <xs:element name="dm" type="dmInfo" />
    </xs:choice>
  </xs:sequence>
 </xs:complexType>

<!-- Definition of a mibTableInfo -->
<xs:complexType name="mibTableInfo">
  <xs:sequence>
   <!-- Mib table name. It will be used for the hook/callback registration. -->
   <xs:element name="name" type="xs:string"/>
   <!-- The fully OID of the mib table (not entry). For instance “1,3,6,1,4,8072,2,2,2“.-->
   <!-- Otherwise, put them in another scalarGroup. -->
   <xs:element name="tableOid" type="xs:string"/>
   <!-- “enabled” is defined in case this group is not ready at back-end yet or deprecated -- >
   <!-- If it’s not enabled, the MIB handler will not load it. It’s enabled if the node doesn’t present (by default) -->
```

```
    <xs:element name="enabled" type="xs:boolean" minOccurs="0"/>
    <!-- Whether the table is writable or not. It's not by default. -->
    <xs:element name="writable" type="xs:boolean" minOccurs="0"/>
    <!-- Specify the maxi entries in the table. Default is 16. -->
    <xs:element name="maxEntries" type="xs:positiveInteger" minOccurs="0"/>
    <!-- Cache is always recommended for performance concerns. The default is 30 seconds -->
     <xs:element name="cacheTimeout" type="xs:nonNegativeInteger" minOccurs="0" />
    <!-- In some special case, the mib table maps to some entries in a DM table. -->
    <!-- The filter may be added here to achieve this goal. For instance, "Routing.%d.type = static" -->
     <xs:element name="mapToEntries" type="xs:string" minOccurs="0" />
    <!-- If callback/hook functions are needed for this group, add it here. -->
     <xs:element name="callbacks" type="callbackInfo " minOccurs="0" />
      <!-- a set of indexs -->
      <xs:element name="index" type="indexInfo" minOccurs="1" maxOccurs="8"/>
      <!-- a set of mibs -->
      <xs:element name="mapping" type="mappingInfo" minOccurs="0" maxOccurs="unbounded"/>
   </xs:sequence>
  </xs:complexType>

<!—Definition of a scalarGroupInfo -->
<xs:complexType name="scalarGroupInfo">
   <xs:sequence>
    <!-- Unique name of group name. It could be the name of baseOid. It will be used for the hook/callback registration. -->
    <xs:element name="name" type="xs:string"/>
    <!-- "baseOid" is the base OID string for this group of scalar mibs. All the mibs have to be in the same OID layer. -->
    <!-- Otherwise, put them in another scalarGroup. -->
     <xs:element name="baseOid" type="xs:string"/>
    <!-- "enabled" is defined in case this group is not ready at back-end yet or deprecated -- >
    <!-- If it's not enabled, the MIB handler will not load it. It's enabled if the node doesn't present (by default) -->
    <xs:element name="enabled" type="xs:boolean" minOccurs="0"/>
    <!-- Cache is always recommended for performance concerns. The default is 30 seconds -->
     <xs:element name="cacheTimeout" type="xs:nonNegativeInteger" minOccurs="0" />
    <!-- In some special case, the scalar mibs may map to an entry in Data Model Table. -->
    <!-- This optional node provides conditional mapping. For instance, "Wifi.Radio.%d.Frequency=2.4" -->
     <xs:element name="mapToEntry" type="xs:string" minOccurs="0" />
    <!-- If callback/hook functions are needed for this group, add it here. -->
     <xs:element name="callbacks" type="callbackInfo " minOccurs="0" />
      <!-- a set of mibs -->
      <xs:element name="mapping" type="mappingInfo" minOccurs="0" maxOccurs="unbounded"/>
   </xs:sequence>
  </xs:complexType>

<xs:complexType name=" mibTableArray">
   <xs:sequence>
    <xs:element name="mibTable" type="mibTableInfo" minOccurs="0" maxOccurs="unbounded" />
```

```
  </xs:sequence>
 </xs:complexType>

<xs:complexType name=" scalarGroupArray ">
  <xs:sequence>
   <xs:element name="scalarGroup" type="scalarGroupInfo" minOccurs="0" maxOccurs="unbounded" />
  </xs:sequence>
 </xs:complexType>

 <!-- Root node defintion. -->
 <xs:element name="mib2DM">
  <xs:complexType>
   <xs:sequence>
    <xs:element name="version" type="xs:positiveInteger"/>
    <xs:element name="name" type="xs:string"/>
    <!-- The last update time. -->
    <xs:element name="lastUpdate" type="xs:string"/>
    <xs:element name="author" type="xs:string"/>
    <!-- Add the library name if callbacks are needed. -->
    <xs:element name="library" type="xs:string" minOccurs="0"/>
    <!-- Scalar Group Array -->
    <xs:element name="scalarGroups" type="scalarGroupArray" minOccurs="0"/>
    <!-- MIB table array -->
    <xs:element name="mibTables" type="mibTableArray" minOccurs="0" />
   </xs:sequence>
  </xs:complexType>
 </xs:element>

</xs:schema>
```

5.7 Mapping XML File Examples

Below is an example of mapping for “clabWIFIWifi” and “ClabWIFISSIDTable” MIB.

```
<?xml version="1.0" encoding="utf-8" ?>
<mib2DM>
    <version>1</version>
    <name>clabWIFIMibs</name>
    <lastUpdate>04/17/2012 </lastUpdate>
    <author>CCSP MIB Team </author>
    <library>CcspMib.o </library>
    <scalarGroups>
       <scalarGroup>
           <name>clabWIFIWiFi</name>
```

```
        <baseOid>1, 3, 6, 1, 4, 1, 4491, 5, 1, 1, 1</baseOid>
        <cacheTimeout>30</cacheTimeout>
        <callbacks>
            <handleRequests>handleWifiScalarRequests</handleRequests>
        </callbacks>
        <mapping>
            <mib>
                <lastOid>1</lastOid>
                <name>clabWIFIWiFiRadioNumberOfEntries</name> <!-- optional -->
                <access>ReadOnly</access>
                <dataType>Unsigned32</dataType>
                <!-- We don't need the range restriction for a ReadOnly mib -->
            </mib>
            <dm>
                <paramName>Device.Wifi.RadioNumberOfEntries</paramName>
                <dataType>unsignedInt</dataType>
            </dm>
        </mapping>
        <mapping>
            <mib>
                <lastOid>2</lastOid>
                <name>clabWIFIWiFiSSIDNumberOfEntries</name> <!-- optional -->
                <access>ReadOnly</access>
                <dataType>Unsigned32</dataType>
                <!-- We don't need the range restriction for a ReadOnly mib -->
            </mib>
            <dm>
                <paramName> Device.Wifi.SSIDNumberOfEntries </paramName>
                <dataType>unsignedInt</dataType>
            </dm>
        </mapping>
        <mapping>
            <mib>
                <lastOid>3</lastOid>
                <name>clabWIFIWiFiAccessPointNumberOfEntries</name> <!-- optional -->
                <access>ReadOnly</access>
                <dataType>Unsigned32</dataType>
                <!-- We don't need the range restriction for a ReadOnly mib -->
            </mib>
            <dm>
                <paramName> Device.Wifi.AccessPointNumberOfEntries</paramName>
                <dataType>unsignedInt</dataType>
            </dm>
        </mapping>
        <mapping>
            <mib>
                <lastOid>4</lastOid>
                <name>clabWIFIWiFiEndPointNumberOfEntries</name> <!-- optional -->
                <access>ReadOnly</access>
                <dataType>Unsigned32</dataType>
                <!-- We don't need the range restriction for a ReadOnly mib -->
            </mib>
            <dm>
                <paramName> Device.Wifi.EndPointNumberOfEntries</paramName>
                <dataType>unsignedInt</dataType>
            </dm>
        </mapping>
    </scalerGroup>
</scalarGroups>
<mibTables>
  <mibTable>
        <name>clabWIFISSIDTable</name>
        <tableOid>1, 3, 6, 1, 4, 1, 4491, 5, 1, 1, 4</tableOid>
```

```
<writable>true</writable>
< mapToEntries>32</mapToEntries>
<cacheTimeout>30</cacheTimeout>
<callbacks>
    <handleRequests>handleWifiTableRequests</handleRequests>
</callbacks>
<index>
    <mib>
        <!-- If index mib is not in this table such as “ifIndex”, set “0” to “lastOid” -->
        <lastOid>0</lastOid>
        <name>ifIndex</name> <!-- optional -->
        <access>NoAccess</access>
        <dataType>InterfaceIndex</dataType>
    </mib>
    <mapToInsNumber>
        <tableObj>Device.WiFiTable. </tableObj>
        <map>
            <from>32</from>
            <to>1</to>
        </map>
        <map>
            <from>112</from>
            <to>2</to>
        </map>
    </mapToInsNumber>
<index>
<index>
    <mib>
        <lastOid>1</lastOid>
        <name>clabWIFISSIDId</name> <!-- optional -->
        <access>NoAccess</access>
        <dataType>InterfaceIndex</dataType>
    </mib>
    <mapToInsNumber>
        <tableObj> Device.WiFiTable.%d.SSID. </tableObj>
        <map>
            <from>32</from>
            <to>1</to>
        </map>
        <map>
            <from>112</from>
            <to>2</to>
        </map>
    </mapToInsNumber>
<index>
<mapping>
    <mib>
        <lastOid>2</lastOid>
        <name>clabWIFISSIDEnable</name> <!-- optional -->
        <access>ReadOnly</access>
        <dataType>TruthValue</dataType>
    </mib>
    <dm>
        <paramName>Device.WiFi.SSID.%d.Enable</paramName>
        <dataType>boolean</dataType>
    </dm>
</mapping>
<mapping>
    <mib>
        <lastOid>3</lastOid>
        <name>clabWIFISSIDDStatus</name> <!-- optional -->
        <access>ReadOnly</access>
        <dataType>INTEGER</dataType>
        <range>
```

```
                    <min>1</min>
                    <max>8</max>
                </range>
            </mib>
            <dm>
                <paramName>Device.WiFi.SSID.%d.Status</paramName>
                <dataType>string</dataType>
                <enumeration>up(1),down(2),unknown(4),dormant(5),notPresent(6),error(8)</enumeration>
            </dm>
        </mapping>
        <mapping>
            <mib>
                <lastOid>4</lastOid>
                <name>clabWIFISSIDAlias</name> <!-- optional -->
                <access>ReadWrite</access>
                <dataType>SnmpAdminString</dataType>
                <range>
                    <min>0</min>
                    <max>64</max>
                </range>
            </mib>
            <dm>
                <paramName>Device.WiFi.SSID.%d.Alias</paramName>
                <dataType>string</dataType>
            </dm>
        </mapping>
        <mapping>
            <mib>
                <lastOid>5</lastOid>
                <name>clabWIFISSIDName</name> <!-- optional -->
                <access>ReadOnly</access>
                <dataType>SnmpAdminString</dataType>
                <range>
                    <min>0</min>
                    <max>64</max>
                </range>
            </mib>
            <dm>
                <paramName>Device.WiFi.SSID.%d.Name</paramName>
                <dataType>string</dataType>
            </dm>
        </mapping>
        <mapping>
            <mib>
                <lastOid>6</lastOid>
                <name>clabWIFISSIDLastChange</name> <!-- optional -->
                <access>ReadOnly</access>
                <dataType>Unsigned32</dataType>
            </mib>
            <dm>
                <paramName>Device.WiFi.SSID.%d.LastChange</paramName>
                <dataType>unsignedInt</dataType>
            </dm>
        </mapping>
        <mapping>
            <mib>
                <lastOid>7</lastOid>
                <name>clabWIFISSIDLowerLayers</name> <!-- optional -->
                <access>ReadWrite</access>
                <dataType>SnmpAdminString</dataType>
                <range>
                    <min>0</min>
                    <max>1024</max>
                </range>
            </mib>
```

```
            <dm>
                <paramName>Device.WiFi.SSID.%d.LowerLayers</paramName>
                <dataType>string</dataType>
            </dm>
        </mapping>
        <mapping>
            <mib>
                <lastOid>8</lastOid>
                <name>clabWIFISSIDBSSID</name> <!-- optional -->
                <access>ReadOnly</access>
                <dataType>MacAddress</dataType>
            </mib>
            <dm>
                <paramName>Device.WiFi.SSID.%d.BSSID</paramName>
                <dataType>string</dataType>
            </dm>
        </mapping>
        <mapping>
            <mib>
                <lastOid>9</lastOid>
                <name>clabWIFISSIDMACAddress</name> <!-- optional -->
                <access>ReadOnly</access>
                <dataType>MacAddress</dataType>
            </mib>
            <dm>
                <paramName>Device.WiFi.SSID.%d.MACAddress</paramName>
                <dataType>string</dataType>
            </dm>
        </mapping>
        <mapping>
            <mib>
                <lastOid>10</lastOid>
                <name>clabWIFISSIDSSID</name> <!-- optional -->
                <access>ReadWrite</access>
                <dataType>SnmpAdminString</dataType>
                <range>
                    <min>0</min>
                    <max>32</max>
                </range>
            </mib>
            <dm>
                <paramName>Device.WiFi.SSID.%d.SSID</paramName>
                <dataType>string</dataType>
            </dm>
        </mapping>
        <mapping>
            <mib>
                <lastOid>11</lastOid>
                <name>clabWIFISSIDRowStatus</name> <!-- optional -->
                <access>ReadWrite</access>
                <dataType>RowStatus</dataType>
            </mib>
            <!-- No DM mapping is needed for RowStatus. -->
        </mapping>

    </mibTable>
  </mibTables>
</mib2DM>
```

End of Document