APPENDIX A

AON Schemas

This appendix contains schemas used by AMC. It includes the following:

Note This information spans multiple pages, making it difficult to select and copy when this guide is viewed in Adobe Acrobat. For best results, go to Cisco.com to view this information on a single page:

http://www.cisco.com/univercd/cc/td/doc/product/aon/admin/amc23/aondb.htm

Archive Schema

When you export a project in AON 2.4, AMC writes a zip file containing configuration data. This configuration archive contains all of the data related to the project. You can then edit the information contained in the archive to modify the configuration data that must change in order for the project to function in the new AMC environment.

The file structure of the configuration archive and schemas for individual data files are listed in the sections that follow. You can use this information to modify configuration files as appropriate for your environment. You can also develop custom scripts to automate this task.

Note Cisco reserves the right to modify these schemas as AON evolves. Schemas are not guaranteed to be backward compatible. Thus custom scripts may need to be adjusted from one release to the next.

```
sample_archive
|-- ArchiveInfo.xml (see ArchiveInfo.xml Schema)
|-- network
|   |-- global
|   |   `-- wccpservers
|   |       `-- 192.168.10.31.xml (see WCCP Server Schema)
|   `-- nodes
|       `-- bangalore
|           |-- NodeInfo.xml (see NodeInfo.xml Schema)
|           |-- acls
|           |   `-- acl1.xml (see ACL Schema)
|           |-- amaproperties
|           |   `-- ha.properties
|           `-- wccpservicegroups
|               `-- 55.xml (see WCCP Service Group Schema)
`-- projects
    |-- shared
    |   `-- System
    |       |-- global
    |       |   |-- extensions (see Extensions)
    |       |   |   |-- bladeletextensions
    |       |   |   |   `-- ABladeletExt.scar
    |       |   |   |-- bladelets
    |       |   |   |   `-- ABladelet.scar
    |       |   |   |-- jmsresourcefiles
    |       |   |   |-- schemas
    |       |   |   |   `-- ASchema.sar
    |       |   |   |-- transformparsers
    |       |   |   |   `-- AParser.xfn
    |       |   |   `-- transforms
    |       |   |   |   `-- ATransform.xfn
    |       |   |-- licenses
    |       |   `-- propertysets
    |       |       `-- com.acme.policies.a
    |       |           `-- 1.0
    |       |               |-- AttributeDomain.xml (see AttributeDomain.xml Schema)
    |       |               |-- ps1.xml (see Property Set Schema)
    |       |               `-- ps2.xml
    |       `-- nodes
    |           `-- bangalore
    |               `-- propertysets
    |                   `-- com.cisco.aons.policies.mec.NextHopDomain
    |                       `-- 1.0
    |                           |-- AttributeDomain.xml
    |                           `-- jms.aontest.com_7600.xml (see Property Set Schema)
    |                           `-- jms.aontest.com_7600.xml.metadata
    `-- standalone
        `-- applProject1
            |-- ProjectInfo.xml
            |-- global
            |   |-- extensions
            |   |   |-- bladeletextensions
            |   |   |-- bladelets
            |   |   |-- jmsresourcefiles
            |   |   |-- schemas
            |   |   |-- transformparsers
            |   |   `-- transforms
            |   `-- propertysets
            `-- nodes
                `-- bangalore
                    |-- flows
                    |   `-- MDS_TEST.MDS_TWOWAY
                    |       |-- META-INF
                    |       |   |-- Flow.xml (see Flow.xml Schema)
```

```
|       |   |-- rules.xml
|       |   `-- MANIFEST.MF
|       `-- com.cisco.aons.ads.userlayout.xml
|-- msgtypes
|   |-- MDS_1W_LrgTTL.xml (see Message Type Schema)
|   |-- MDS_2W_LrgTTL.xml
|   `-- msgtype_order.xml (see Message Type Order File Schema)
`-- propertysets
    `-- com.cisco.aons.policies.mec
        `-- 1.0
            |-- AttributeDomain.xml Schema
            |-- mds-dst-jms.aontest.com_7666.xml
            |   (see Property Set Schema)
            `-- mds-dst-jms.aontest.com_7666.xml.metadata
```

ArchiveInfo.xml Schema

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <xsd:element name="ArchiveInfo">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="SourceAMCInfo" type="AMCInfo" />
            </xsd:sequence>
            <xsd:attribute name="formatVersion" type="xsd:string"
                default="1.0">
            </xsd:attribute>
            <xsd:attribute name="createdAt" type="xsd:dateTime"
                use="optional">
            </xsd:attribute>
        </xsd:complexType>
    </xsd:element>

    <xsd:complexType name="AMCInfo">
        <xsd:attribute name="hostname" type="xsd:string"
            use="required">
        </xsd:attribute>
        <xsd:attribute name="aonVersionString" type="xsd:string"
            use="required">
        </xsd:attribute>
    </xsd:complexType>
</xsd:schema>
```

NodeInfo.xml Schema

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <xsd:element name="NodeInfo">
        <xsd:complexType>
            <xsd:choice>
                <xsd:element ref="StandaloneNode"></xsd:element>
                <xsd:element ref="VirtualClusterNode"></xsd:element>
            </xsd:choice>
            <xsd:attribute name="schemaVersion" type="xsd:string"
                default="1.0">
            </xsd:attribute>
        </xsd:complexType>
    </xsd:element>

    <xsd:element name="StandaloneNode">
        <xsd:complexType>
```

```
            <xsd:attribute name="platform" type="xsd:string" use="required">
            </xsd:attribute>
            <xsd:attribute name="name" type="xsd:string"
                use="required"/>
            <xsd:attribute name="aonVersionString" type="xsd:string"
                use="required"/>
        </xsd:complexType>
    </xsd:element>

    <xsd:element name="VirtualClusterNode">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element ref="StandaloneNode"
maxOccurs="unbounded" minOccurs="1"/>
            </xsd:sequence>
            <xsd:attribute name="name" type="xsd:string"
                use="required"/>
        </xsd:complexType>
    </xsd:element>
</xsd:schema>
```

ProjectInfo.xml Schema

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <xsd:element name="ProjectInfo">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="Description" type="xsd:string" />
                <xsd:element ref="Users" />
                <xsd:element ref="Nodes" />
            </xsd:sequence>
            <xsd:attribute name="schemaVersion" type="xsd:string"
                default="1.0" />

            <xsd:attribute name="name" type="xsd:string" use="required" />

            <xsd:attribute name="prefix" type="xsd:string"
                use="required" />

            <xsd:attribute name="shared" type="xsd:boolean"
                use="required" />

        </xsd:complexType>
    </xsd:element>

    <xsd:element name="Users">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element ref="User" minOccurs="0" maxOccurs="unbounded"/>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name="User">
        <xsd:complexType>
            <xsd:attribute name="name" type="xsd:string" use="required" />

            <xsd:attribute name="realm" type="xsd:string"
                default="AMCLocal" />

        </xsd:complexType>
    </xsd:element>
```

```
    <xsd:element name="Nodes">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element ref="Node" minOccurs="0" maxOccurs="unbounded"/>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name="Node">
        <xsd:complexType>
            <xsd:attribute name="name" type="xsd:string" use="required" />
        </xsd:complexType>
    </xsd:element>

</xsd:schema>
```

Node Mapping File Schema

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
     elementFormDefault="qualified">

    <xsd:element name="PromotionNodeMap">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element ref="SourceNode" maxOccurs="unbounded" />
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>

    <xsd:element name="SourceNode">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element ref="DestinationNode"
                   minOccurs="0" maxOccurs="unbounded"/>
            </xsd:sequence>
            <xsd:attribute name="name" type="xsd:string"/>
        </xsd:complexType>
    </xsd:element>

    <xsd:element name="DestinationNode">
        <xsd:complexType>
            <xsd:attribute name="name" type="xsd:string"/>
        </xsd:complexType>
    </xsd:element>
</xsd:schema>
```

Flow.xml Schema

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
  elementFormDefault="qualified"
  attributeFormDefault="unqualified">
    <xs:element name="flow">
        <xs:annotation>
            <xs:documentation>
               Root element for describing a flow
             </xs:documentation>
        </xs:annotation>
        <xs:complexType>
            <xs:sequence>
                <xs:element ref="flow-vars" minOccurs="0"/>
```

```
                <xs:element name="flow-init" minOccurs="0">
                    <xs:complexType>
                        <xs:sequence>
                            <xs:element ref="operator-blocks" minOccurs="0"/>
                        </xs:sequence>
                    </xs:complexType>
                </xs:element>
                <xs:element name="flow-steps">
                    <xs:complexType>
                        <xs:sequence>
                            <xs:elementname="request-action"
                                type="actionType"/>
                            <xs:elementname="response-action"
                                type="actionType"
                                minOccurs="0" maxOccurs="unbounded"/>
                            <xs:element name="compensation-steps"
                                type="actionType" minOccurs="0"/>
                        </xs:sequence>
                    </xs:complexType>
                </xs:element>
                <!-- Scope Global scope is necessary -->
                <xs:element ref="scope-blocks" minOccurs="1"/>
                <xs:element ref="operator-blocks" minOccurs="0"/>
            </xs:sequence>
            <xs:attribute name="name" type="xs:string" use="required"/>
            <xs:attribute name="package" type="xs:string" use="optional"/>
            <xs:attribute name="description" type="xs:string" use="required"/>
            <xs:attribute name="interactionStyle" type="interactionType"
               use="required"/>
            <xs:attribute name="logging" type="xs:string"
               use="optional" default="none"/>
            <xs:attribute name="sla" type="xs:unsignedInt"
               use="optional" default="0"/>
            <xs:attribute name="enableRM" type="xs:boolean"
               use="optional" default="false"/>
            <xs:attribute name="compress" type="xs:boolean"
               use="optional" default="false"/>
            <!-- Scope (Global) ScopeId is optional for flow -->
            <xs:attribute name="scopeId" type="xs:string" use="optional"/>
            <xs:attribute name="opId" type="xs:string" use="optional"/>
        </xs:complexType>
    </xs:element>
      <xs:element name="flow-vars">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="flow-var"
                     minOccurs="0" maxOccurs="unbounded">
                    <xs:complexType>
                        <xs:attribute name="type" type="xs:string"
                            use="required"/>
                        <xs:attribute name="name" type="xs:string"
                            use="required"/>
                        <xs:attribute name="id" type="xs:string"/>
                    </xs:complexType>
                </xs:element>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
    <xs:simpleType name="interactionType">
        <xs:annotation>
            <xs:documentation>
               Defines the interaction type for the flow
             </xs:documentation>
        </xs:annotation>
```

```
    <xs:restriction base="xs:string">
        <xs:enumeration value="Request-Response"/>
        <xs:enumeration value="Request-Only"/>
    </xs:restriction>
</xs:simpleType>
<xs:complexType name="actionType">
    <xs:sequence>
        <xs:element name="first-bladelet" minOccurs="0">
            <xs:complexType>
                <xs:attribute name="id" type="xs:string" use="required"/>
            </xs:complexType>
        </xs:element>
        <xs:element ref="bladelet" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="id" type="xs:string" use="optional"/>
    <!-- Scope ScopeId is optional for actionType -->
    <xs:attribute name="scopeId" type="xs:string" use="optional"/>
</xs:complexType>
<xs:element name="bladelet">
    <xs:complexType>
        <xs:sequence>
            <xs:element ref="param-values"
               minOccurs="0" maxOccurs="unbounded"/>
            <xs:element ref="exported-value"
               minOccurs="0" maxOccurs="unbounded"/>
            <xs:element ref="next-steps" minOccurs="0"/>
        </xs:sequence>
        <xs:attribute name="name" type="xs:string" use="required"/>
        <xs:attribute name="UUID" type="xs:string" use="optional"/>
        <xs:attribute name="description" type="xs:string" use="required"/>
        <xs:attribute name="id" type="xs:string" use="required"/>
        <xs:attribute name="opBlockId" type="xs:string" use="optional"/>
        <!-- Scope ScopeId is required for bladelet -->
        <xs:attribute name="scopeId" type="xs:string" use="required"/>
    </xs:complexType>
</xs:element>
<xs:element name="operator-blocks">
    <xs:complexType>
        <xs:sequence>
            <xs:element name="operator-block"
                 minOccurs="0" maxOccurs="unbounded">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="assignment"
                              minOccurs="0" maxOccurs="unbounded">
                            <xs:complexType mixed="true">
                                <xs:attribute name="from"
                                    type="xs:string" use="required"/>
                                <xs:attribute name="to"
                                    type="xs:string" use="required"/>
                                <xs:attribute name="fromType"
                                    type="xs:string" use="required"/>
                                <xs:attribute name="toType"
                                    type="xs:string" use="required"/>
                            </xs:complexType>
                        </xs:element>
                    </xs:sequence>
                    <xs:attribute name="id"type="xs:string"
                        use="required"/>
                    <xs:attribute name="name" type="xs:string"
                        use="optional"/>
                    <!-- Scope ScopeId is optional for operator blocks -->
                    <xs:attribute name="scopeId" type="xs:string"
                        use="optional"/>
```

```
                </xs:complexType>
            </xs:element>
        </xs:sequence>
    </xs:complexType>
</xs:element>
  <!-- Scope ScopeBlock id and parent same for global one -->
<xs:element name="scope-blocks">
    <xs:complexType>
        <xs:sequence>
            <xs:element name="scope" minOccurs="0" maxOccurs="unbounded">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element ref="flow-vars"/>
                    </xs:sequence>
                    <xs:attribute name="id"type="xs:string"
                        use="required"/>
                    <xs:attribute name="name" type="xs:string"
                        use="optional"/>
                    <xs:attribute name="parent" type="xs:string"
                        use="required"/>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
    </xs:complexType>
</xs:element>
<xs:element name="list-element">
    <xs:complexType>
        <xs:choice>
            <xs:element ref="value"/>
            <xs:element ref="param-values" maxOccurs="unbounded"/>
        </xs:choice>
        <xs:attribute name="type" type="xs:string"
           use="optional" default="string"/>
        <xs:attribute name="elementClass" type="xs:string" use="optional"/>
    </xs:complexType>
</xs:element>
<xs:element name="param-values">
    <xs:complexType>
        <xs:choice>
            <xs:element ref="value" minOccurs="0"/>
            <xs:element ref="list-element"
               minOccurs="0" maxOccurs="unbounded"/>
            <xs:element ref="param-values"
               minOccurs="0" maxOccurs="unbounded"/>
        </xs:choice>
        <xs:attribute name="name" type="xs:string" use="required"/>
        <xs:attribute name="designName" type="xs:string" use="optional"/>
        <xs:attribute name="type" type="xs:string"
            use="optional" default="string"/>
        <xs:attribute name="paramGroupFQN" type="xs:string"
            use="optional" default="string"/>
        <xs:attribute name="keyName" type="xs:string" use="optional"/>
        <xs:attribute name="elementClass" type="xs:string" use="optional"/>
        <xs:attribute name="bindId" type="xs:string" use="optional"/>
        <xs:attribute name="bindType" type="xs:string" use="optional"/>
        <xs:attribute name="output" type="xs:boolean" use="optional"/>
    </xs:complexType>
</xs:element>
<xs:element name="exported-value">
    <xs:complexType>
        <xs:attribute name="type" type="xs:string" use="required"/>
        <xs:attribute name="name" type="xs:string" use="required"/>
        <xs:attribute name="designName" type="xs:string" use="optional"/>
        <xs:attribute name="bindId" type="xs:string" use="required"/>
```

```
            </xs:complexType>
        </xs:element>
        <xs:element name="next-steps">
            <xs:complexType>
                <xs:sequence>
                    <xs:element name="next-step"
                         minOccurs="0" maxOccurs="unbounded">
                        <xs:complexType>
                            <xs:attribute name="label" type="xs:string"
                                use="optional"/>
                            <xs:attribute name="id"type="xs:string"
                                use="optional"/>
                            <xs:attribute name="opBlockId" type="xs:string"
                                use="optional"/>
                            <xs:attribute name="isBreak" type="xs:boolean"
                                use="optional"/>
                        </xs:complexType>
                    </xs:element>
                    <xs:element name="response-link" minOccurs="0">
                        <xs:complexType>
                            <xs:attribute name="label" type="xs:string"
                                use="optional"/>
                            <xs:attribute name="id"type="xs:string"
                                use="optional"/>
                            <xs:attribute name="opBlockId" type="xs:string"
                                use="optional"/>
                            <xs:attribute name="isBreak" type="xs:boolean"
                                use="optional"/>
                        </xs:complexType>
                    </xs:element>
                    <xs:element name="on-exception"
                         minOccurs="0" maxOccurs="unbounded">
                        <xs:complexType>
                            <xs:attribute name="exId" type="xs:string"
                                use="optional" default="default"/>
                            <xs:attribute name="id" type="xs:string"
                                use="optional"/>
                            <xs:attribute name="opBlockId" type="xs:string"
                                use="optional"/>
                            <xs:attribute name="isBreak" type="xs:boolean"
                                use="optional"/>
                        </xs:complexType>
                    </xs:element>
                </xs:sequence>
            </xs:complexType>
        </xs:element>
        <xs:element name="value" type="xs:string"/>
</xs:schema>
```

Message Type Schema

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
  elementFormDefault="qualified">
  <xs:element name="msgType">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="fTuple"/>
        <xs:element ref="uri"/>
        <xs:element ref="params"/>
        <xs:element ref="headers"/>
        <xs:element ref="content"/>
```

```
      <xs:element ref="policyPart"/>
    </xs:sequence>
    <xs:attribute name="name" use="required" type="xs:NCName"/>
  </xs:complexType>
</xs:element>
<xs:element name="fTuple">
  <xs:complexType>
    <xs:attribute name="name" use="required" type="xs:NCName"/>
  </xs:complexType>
</xs:element>
<xs:element name="uri">
  <xs:complexType>
    <xs:attribute name="pattern" use="required"/>
  </xs:complexType>
</xs:element>
<xs:element name="params">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="param"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="param">
  <xs:complexType>
    <xs:attribute name="name" use="required" type="xs:NCName"/>
    <xs:attribute name="op" use="required" type="xs:NCName"/>
    <xs:attribute name="value" use="required" type="xs:integer"/>
  </xs:complexType>
</xs:element>
<xs:element name="headers">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="header"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="header">
  <xs:complexType>
    <xs:attribute name="name" use="required" type="xs:NCName"/>
    <xs:attribute name="op" use="required" type="xs:NCName"/>
    <xs:attribute name="value" use="required" type="xs:integer"/>
  </xs:complexType>
</xs:element>
<xs:element name="content">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="expr"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="expr">
  <xs:complexType>
    <xs:attribute name="op" use="required" type="xs:NCName"/>
    <xs:attribute name="value" use="required" type="xs:NCName"/>
    <xs:attribute name="xpath" use="required"/>
  </xs:complexType>
</xs:element>
<xs:element name="policyPart">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="flow"/>
      <xs:element ref="policy"/>
    </xs:sequence>
  </xs:complexType>
```

```
  </xs:element>
  <xs:element name="flow">
    <xs:complexType>
      <xs:attribute name="flowId" use="required" type="xs:NCName"/>
    </xs:complexType>
  </xs:element>
  <xs:element name="policy">
    <xs:complexType>
      <xs:sequence>
        <xs:element maxOccurs="unbounded" ref="attr"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="attr">
    <xs:complexType>
      <xs:attribute name="domain" use="required"/>
      <xs:attribute name="propertyset" use="required"/>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Message Type Order File Schema

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <xsd:element name="MsgTypesOrder">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element ref="MsgType"
                    maxOccurs="unbounded" minOccurs="1"/>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>

    <xsd:element name="MsgType">
        <xsd:complexType>
            <xsd:attribute name="name" type="xsd:string"
                use="required"/>
        </xsd:complexType>
    </xsd:element>

</xsd:schema>
```

AttributeDomain.xml Schema

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" version="1.0">
    <xs:element name="AttributeDomain">
        <xs:complexType>
            <xs:sequence>
                <xs:element ref="AttributeDefinition"
                    minOccurs="0" maxOccurs="unbounded"/>
                <xs:element ref="ForwardReference" minOccurs="0"/>
                <xs:element ref="InverseReference" minOccurs="0"/>
                <xs:element ref="KeyDefinition" minOccurs="0"/>
            </xs:sequence>
            <xs:attribute name="desc" type="xs:string" use="optional"/>
            <xs:attribute name="domainName" type="xs:NMTOKEN" use="required"/>
            <xs:attribute name="fixed" type="xs:boolean"
                use="optional" default="false"/>
            <xs:attribute name="schemaVersion" type="xs:string"
```

```
            use="optional" default="1.0"/>
        <xs:attribute name="version" type="xs:string"
            use="optional" default="1.0"/>
        <xs:attribute name="policyGroup" type="xs:string" use="optional"/>
    </xs:complexType>
</xs:element>
<xs:element name="AttributeDefinition">
    <xs:complexType>
        <xs:sequence>
            <xs:element ref="PossibleValue"
            minOccurs="0" maxOccurs="unbounded"/>
            <xs:element ref="DefaultValue"
            minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:attribute name="passwordfield" type="xs:boolean"
            use="optional" default="false"/>
        <xs:attribute name="readonly" type="xs:boolean"
            use="optional" default="false"/>
        <xs:attribute name="psref" type="xs:NMTOKEN" use="optional"/>
        <xs:attribute name="attributeName" type="xs:string"
            use="required"/>
        <xs:attribute name="type" type="attributeType"
            use="optional" default="string"/>
        <xs:attribute name="visible" type="xs:boolean"
            use="optional" default="true"/>
        <xs:attribute name="desc" type="xs:string" use="optional"/>
        <xs:attribute name="tooltipKey" type="xs:string" use="optional"/>
        <xs:attribute name="nullable" type="xs:boolean"
            use="optional" default="false"/>
        <xs:attribute name="minValue" type="xs:integer" use="optional"/>
        <xs:attribute name="maxValue" type="xs:integer" use="optional"/>
        <xs:attribute name="maxLength" type="xs:integer" use="optional"/>
        <xs:attribute name="externalDataClass" type="xs:string"
            use="optional"/>
    </xs:complexType>
</xs:element>
<xs:element name="PossibleValue">
    <xs:complexType>
        <xs:attribute name="value" type="xs:string" use="required"/>
    </xs:complexType>
</xs:element>
<xs:element name="DefaultValue">
    <xs:complexType>
        <xs:attribute name="value" type="xs:string" use="required"/>
    </xs:complexType>
</xs:element>
<xs:element name="ForwardReference" type="xs:string"/>
<xs:element name="InverseReference" type="xs:string"/>
<xs:element name="KeyDefinition">
    <xs:complexType>
        <xs:sequence>
            <xs:element ref="AttributeReference"
                minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>
</xs:element>
<xs:element name="AttributeReference">
    <xs:complexType>
        <xs:attribute name="name" type="xs:string" use="required"/>
    </xs:complexType>
</xs:element>

<xs:simpleType name="attributeType">
    <xs:restriction base="xs:string">
```

```
            <xs:enumeration value="string"/>
            <xs:enumeration value="enum"/>
            <xs:enumeration value="list"/>
            <xs:enumeration value="integer"/>
            <xs:enumeration value="ipv4host"/>
            <xs:enumeration value="port"/>
            <!--
            <xs:enumeration value="password"/>
            <xs:enumeration value="boolean"/>
            <xs:enumeration value="url"/>
            <xs:enumeration value="uriList"/>
            <xs:enumeration value="jdbcUrl"/>
            -->
        </xs:restriction>
    </xs:simpleType>
</xs:schema>
```

Property Set Schema

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
  version="1.0">
    <xs:element name="PropertySet">
        <xs:complexType>
            <xs:sequence>
                <xs:element ref="Attribute"
                    minOccurs="0" maxOccurs="unbounded"/>
            </xs:sequence>
            <xs:attribute name="key" type="xs:string" use="required"/>
            <xs:attribute name="version" type="xs:string" use="optional" default="1.0"/>
        </xs:complexType>
    </xs:element>
    <xs:element name="Attribute">
        <xs:complexType>
            <xs:sequence>
                <xs:element ref="Value"
                    minOccurs="0" maxOccurs="unbounded"/>
            </xs:sequence>
            <xs:attribute name="name" type="xs:string" use="required"/>
        </xs:complexType>
    </xs:element>
    <xs:element name="Value">
        <xs:complexType>
            <xs:attribute name="value" type="xs:string" use="required"/>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

ACL Schema

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
  elementFormDefault="qualified">
  <xs:element name="fTuple">
    <xs:complexType>
        <xs:attribute name="match" type="xs:string"/>
        <xs:attribute name="srcIp" type="xs:string"/>
        <xs:attribute name="srcPort" type="xs:string"/>
        <xs:attribute name="srcMask" type="xs:string"/>
        <xs:attribute name="destIp" type="xs:string"/>
        <xs:attribute name="destPort" type="xs:string"/>
```

```
        <xs:attribute name="destMask" type="xs:string"/>
      <xs:attribute name="name" use="required" type="xs:string"/>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Extensions

Extensions are exported in the same format as when they were originally uploaded in AMC.

License

Licenses are exported in the same format as when they were originally uploaded in AMC.

WCCP Service Group Schema

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
  elementFormDefault="qualified">
  <xs:element name="WCCPConfig">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="ServiceGroup"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="ServiceGroup">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="InterceptionPorts"/>
        <xs:element ref="ApplicationListenPort"/>
        <xs:element ref="fTuple" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element ref="Client"/>
        <xs:element ref="Server" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
      <xs:attribute name="forClusterMgmt" use="required" type="xs:NCName"/>
      <xs:attribute name="id" use="required" type="xs:integer"/>
      <xs:attribute name="loadBalancingRule" use="required"
        type="xs:integer"/>
      <xs:attribute name="multicastGroup" use="required" type="xs:NMTOKEN"/>
      <xs:attribute name="status" use="required" type="xs:NCName"/>
      <xs:attribute name="authenticationPwd" type="xs:NCName"/>
    </xs:complexType>
  </xs:element>
  <xs:element name="InterceptionPorts" type="xs:string"/>
  <xs:element name="ApplicationListenPort" type="xs:integer"/>
  <xs:element name="fTuple">
    <xs:complexType>
      <xs:attribute name="name" use="required"/>
    </xs:complexType>
  </xs:element>
  <xs:element name="Client">
    <xs:complexType>
      <xs:attribute name="cn" use="required"/>
    </xs:complexType>
  </xs:element>
  <xs:element name="Server">
    <xs:complexType>
      <xs:sequence>
```

```
        <xs:element ref="GroupListenInterfaces"/>
        <xs:element ref="RedirectInterfaces"/>
      </xs:sequence>
      <xs:attribute name="ipAddress" use="required" type="xs:NMTOKEN"/>
    </xs:complexType>
  </xs:element>
  <xs:element name="GroupListenInterfaces" type="Interfaces"/>
  <xs:element name="RedirectInterfaces" type="Interfaces"/>
  <xs:complexType name="Interfaces">
    <xs:sequence>
      <xs:element maxOccurs="unbounded" ref="Interface"/>
    </xs:sequence>
  </xs:complexType>
  <xs:element name="Interface">
    <xs:complexType>
      <xs:attribute name="name" use="required" type="xs:NCName"/>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

WCCP Server Schema

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
  elementFormDefault="qualified">
  <xs:element name="Server">
    <xs:complexType>
        <xs:attribute name="accessMethod" type="xs:string"/>
        <xs:attribute name="enablePassword" type="xs:string"/>
        <xs:attribute name="ipAddress" type="xs:string"/>
        <xs:attribute name="password" type="xs:string"/>
        <xs:attribute name="status" type="xs:string"/>
        <xs:attribute name="userName" type="xs:string"/>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Programmatic Management Interface APIs

The programmatic management interface feature (PMI) provides an interface so that third-party applications can manipulate data in AMC. This feature operates under the following assumptions:

- The client must use HTTPS for transport security to connect to AMC. The client must have a server certificate in the AMC trust store to properly authenticate.
- The client must use the “aonsadmin” user ID to log into AMC.
- All the web service calls are RPC based.
- SOAP version 1.1 is supported. Version 1.2 is not supported.
- With the exception of loginUser, all error messages are handled as SOAPFault. The client is expected to process the SOAPFault appropriately.

The following APIs are included:

- loginUser API, page A-16
- listOfProjects API, page A-17
- exportProject API, page A-18

A sample PMI client is included with the installation of AMC 2.4. See the "Sample PMI Client" section on page A-25 for further details.

loginUser API

This operation allows the PMI client to login and use the subsequent Web service operations. The client is expected use the "aonsadmin" user for PMI Web service operations.

Input Parameters

- loginRequest element contains the following:
  - name: userid
  - password: user's password
  - versionId: future versioning perspective

Output Parameters:

- loginResponse element contains the following:
  - name: userid
  - errorCode: 0 for success. Non-zero values for errors
  - errorMessage: optional error message element

Schema

```
<xs:complexType name="LoginRequestType">
    <xs:sequence>
        <xs:element ref="amcData" minOccurs="1" maxOccurs="1"/>
    </xs:sequence>
</xs:complexType>

<xs:element name="amcData">
    <xs:complexType>
        <xs:sequence>
            <xs:element ref="login" minOccurs="1" maxOccurs="1"/>
        </xs:sequence>
    </xs:complexType>
</xs:element>

<xs:element name="login">
    <xs:complexType>
        <xs:attribute name="name" type="xs:string" use="required"/>
        <xs:attribute name="password" type="xs:string" use="required"/>
        <xs:attribute name="versionId" type="xs:string" use="required"/>
    </xs:complexType>
</xs:element>

<xs:complexType name="LoginResponseType">
    <xs:sequence>
        <xs:element ref="amcDataResult" minOccurs="1" maxOccurs="1"/>
    </xs:sequence>
```

```
</xs:complexType>

<xs:element name="amcDataResult">
    <xs:complexType>
        <xs:sequence>
            <xs:element ref="login" minOccurs="1" maxOccurs="1"/>
        </xs:sequence>
        <xs:attribute name="amcVersion" type="xs:string" />
    </xs:complexType>
</xs:element>

<xs:element name="login">
    <xs:complexType>
        <xs:attribute name="name" type="xs:string" use="required"/>
        <xs:attribute name="errorCode" type="xs:string" use="required"/>
        <xs:attribute name="errorMessage" type="xs:string" />
    </xs:complexType>
</xs:element>
```

listOfProjects API

This operation lists all of the projects from the AMC source environment. Later clients will iterate through the project list and may invoke export operation.

Input Parameters

- listOfProjectsRequest element contains the following:
  - /* Returns always the available projects with status */

Output Parameters

- listOfProjectsResponse element contains the following:
  - list of project details (project names)

Schema

```
<xs:complexType name="ListOfProjectsRequestType">
</xs:complexType>

<xs:complexType name="ListOfProjectsResponseType">
    <xs:complexType>
        <xs:sequence>
            <xs:element  ref ="projectdetails" minOccurs="0"/>
        </xs:sequence>
    </xs:complexType>
</xs:complexType>

<xs:element name="projectdetail">
    <xs:complexType>
        <xs:attribute name="name" type="xs:string" use="required"/>
    </xs:complexType>
</xs:element>

<xs:element name="projectdetails">
    <xs:complexType>
        <xs:sequence>
            <xs:element ref="projectdetail"  minOccurs="1" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>
```

```
</xs:element>
```

exportProject API

This operation exports the resources for a project in a given location.

Input Parameters:

- exportRequest element contains the following:
  - type: string (Type of the entity to be exported. Supported type is project)
  - list of names (name uniquely identify the entity such as project)
  - filename: string (Archive file name)
  - location: string (Location to which the archive file will be stored for export or from which the file will be retrieved.)

Output Parameters:

- exportResponse element contains the following:
  - status: string (success or fail)

Schema

```
<xs:complexType name="ExportRequestType">
    <xs:complexType>
        <xs:sequence>
            <xs:element name="type"  />
            <xs:element ref="projects"  minOccurs="0" />
            <!-- if there is no projects element, then it is all projects -->
            <xs:element name="filename" />
            <xs:element name="location" />
        </xs:sequence>
    </xs:complexType>
</xs:complexType>

<xs:element name="project">
    <xs:complexType>
        <xs:attribute name="name" type="xs:string" use="required"/>
    </xs:complexType>
</xs:element>

<xs:element name="projects">
    <xs:complexType>
        <xs:sequence>
            <xs:element ref="project"  minOccurs="1" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>
</xs:element>

<xs:complexType name="ExportResponseType">
    <xs:complexType>
        <xs:sequence>
            <xs:element name="status">
                <xs:simpleType>
                    <xs:restriction base="xs:token">
                        <xs:enumeration value="success"/>
                        <xs:enumeration value="fail"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:element>
        </xs:sequence>
```

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

importProject API

This operation imports a project and its resources from the given location.

Input Parameters

- importRequest element contains the following:
  - type: string (Type of the entity to be imported. Supported type is project)
  - filename: string (Archive file name)
  - location: string (Location from which the archive file will be retrieved for import)
  - mappingfile: string (name of mapping file)
  - mappinglocation: string (location of mapping file)
  - import type (possible values are add or merge.)

Output Parameters

- ImportResponse element contains the following:
  - status: string (success or fail)

Schema

```
<xs:complexType name="ImportRequestType">
    <xs:complexType>
        <xs:sequence>
            <xs:element name="type"  />
            <xs:element name="filename"  />
            <xs:element name="location"  />
            <xs:element name="nodemappingfile"/>
            <xs:element name="mappinglocation"/>
            <xs:element name="importtype">
                <xs:simpleType>
                    <xs:restriction base="xs:token">
                        <xs:enumeration value="ADDON"/>
                        <xs:enumeration value="REPLACE"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:element>
        </xs:sequence>
    </xs:complexType>
</xs:complexType>

<xs:complexType name="ImportResponseType">
    <xs:complexType>
        <xs:sequence>
            <xs:element name="status">
                <xs:simpleType>
                    <xs:restriction base="xs:token">
                        <xs:enumeration value="success"/>
                        <xs:enumeration value="fail"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:element>
        </xs:sequence>
```

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

listOfDRs API

This operation lists all the DRs for a given project. Later clients will iterate through the DR and invoke deploy operation.

Input Parameters:

- listOfDRRequest element contains the following:
  - type: string (Type of the entity (for which DRs created). Supported type is project.)
  - name: string (Uniquely identify the entity such as project ID). If this element is not present, default is all projects.
  - statustype: string (type of DR such as created, staged, deployed). If this element is not present, default is all status types.

Output Parameters:

- listOfDRResponse element contains the following:
  - list of drs (DR ID and statustype)
  - status: string (success or fail)

Schema

```
<xs:complexType name="listOfDRRequestType">
    <xs:complexType>
        <xs:sequence>
            <xs:element name="type"  />
            <xs:element name="name" />
            <xs:element name="statustype">
                <xs:simpleType>
                    <xs:restriction base="xs:token">
                        <xs:enumeration value="Saved"/>
                        <xs:enumeration value="Staged"/>
                        <xs:enumeration value="Deployed"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:element>
        </xs:sequence>
    </xs:complexType>
</xs:complexType>

<xs:complexType name="ListOfDRResponseType">
    <xs:complexType>
        <xs:sequence>
            <xs:element  ref="drs" />
        </xs:sequence>
    </xs:complexType>
</xs:complexType>

<xs:element name="dr">
    <xs:complexType>
        <xs:attribute name="id" type="xs:string" use="required"/>
        <xsd:attribute name="nodeName" type="xsd:string" use="optional"/>
        <xsd:attribute name="statusType" type="xsd:string" use="required"/>
        <xsd:attribute name="created" type="xsd:string" use="required"/>
        <xsd:attribute name="project" type="xsd:string" use="required"/>
```

```
    </xs:complexType>
</xs:element>

<xs:element name="drs">
    <xs:complexType>
        <xs:sequence>
            <xs:element ref="dr"  minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>
</xs:element>
```

deployDR API

This operation used to stage and deploy the given DR for a particular project.

Input Parameters:

- deployRequest element contains the following:
  - drid: string (Uniquely identify the DR within AMC)

Output Parameters:

- deployResponse element contains the following:
  - status: string (success or fail)

Schema

```
<xs:complexType name="drAttributesType">
    <xs:attribute name="id" type="xs:string" use="required"/>
</xs:complexType>


<xs:complexType name="drsType" mixed="false">
    <xs:annotation>
        <xs:documentation xml:lang="en">

            Implementation only supports a single deployment request
            ID per SOAP request message.

        </xs:documentation>
    </xs:annotation>
    <xs:sequence>
        <xs:element name="dr" type="drAttributesType"
            minOccurs="1" maxOccurs="1"/>
        </xs:sequence>

    </xs:complexType>

    <xs:complexType name="DeployRequestType">
        <xs:annotation>
            <xs:documentation xml:lang="en">

                Implementation only supports a single drs tag element
                per SOAP request message.

            </xs:documentation>
        </xs:annotation>
        <xs:sequence>
            <xs:element name="drs" type="drsType"
                minOccurs="1" maxOccurs="1"/>
```

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

        <xs:element name="deployDR" type="DeployRequestType"/>

    </xs:schema>


    <xs:complexType name="DeployResponseType">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="status" >
                    <xs:simpleType>
                        <xs:restriction base="xs:token">
                            <xs:enumeration value="success"/>
                            <xs:enumeration value="fail"/>
                        </xs:restriction>
                    </xs:simpleType>
                </xs:element>
            </xs:sequence>
        </xs:complexType>
    </xs:complexType>
```

exportAccessControlMapping API

This operation exports the users, permissions, user-roles, role-permissions from a source environment to an archive file.

There will be one XML file that will contain all the mapping information. Refer AccessControlMappingFileStructure type for details.

Input Parameters

- exportAccessControlMapping element contains the following:
  - type: string

    all is for users, permissions, user-roles, role-permissions

    user is for users, user-roles

    permission is for permissions, roles-permissions
  - filename: string (Archive file name)
  - location: string (Location to which the archive file will be stored for export)

Output Parameters

- accessControlMappingExportResponse element contains the following:
  - status: string (success or fail)

Schema

```
<xs:complexType name="userAttributesType" mixed="true">
    <xs:attribute name="id" type="xs:integer" use="required"/>
    <xs:attribute name="first_name" type="xs:string" use="required"/>
    <xs:attribute name="last_name" type="xs:string" use="required"/>
    <xs:attribute name="login_id" type="xs:string" use="required"/>
    <xs:attribute name="email_address" type="xs:string" use="required"/>
    <xs:attribute name="password" type="xs:string" use="required"/>
    <xs:attribute name="realm" type="xs:string" use="required"/>
    <xs:attribute name="active_flag" type="xs:string" use="optional"/>
```

```
    <xs:attribute name="inactivated_date" type="xs:date" use="optional"/>
</xs:complexType>

<xs:complexType name="userType" mixed="true">
    <xs:complexContent>
        <xs:extension base="userAttributesType">
            <xs:sequence>
                <xs:element name="roles" type="rolesType"
                    minOccurs="1" maxOccurs="unbounded"/>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>


    <xs:complexType name="usersType">
        <xs:sequence>
            <xs:element name="user" type="userType" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="roleType" mixed="true">
        <xs:attribute name="id" type="xs:integer" use="required"/>
        <xs:attribute name="name" type="xs:string" use="required"/>
    </xs:complexType>


    <xs:complexType name="rolesType">
        <xs:sequence>
            <xs:element name="role" type="roleType" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="rolePermissionAttrType">
        <xs:attribute name="id" type="xs:integer" use="required"/>
        <xs:attribute name="name" type="xs:string" use="required"/>
    </xs:complexType>

    <!-- This element contains permissions on a specified resource type (ex: PEP,
MessageType) -->
    <xs:complexType name="permissionAttrType" mixed="true">

        <!-- id is system generated identifier. This shouldn't be modified during export
and import -->

        <xs:attribute name="id" type="xs:integer" use="required"/>
        <xs:attribute name="activate" type="xs:integer" use="required"/>
        <xs:attribute name="create" type="xs:integer" use="required"/>
        <xs:attribute name="delete" type="xs:integer" use="required"/>
        <xs:attribute name="deploy" type="xs:integer" use="required"/>
        <xs:attribute name="export" type="xs:integer" use="required"/>
        <xs:attribute name="import" type="xs:integer" use="required"/>
        <xs:attribute name="name" type="xs:string" use="required"/>
        <xs:attribute name="read" type="xs:integer" use="required"/>
        <xs:attribute name="replace" type="xs:integer" use="required"/>
        <!-- resource_type is identifier of a resource type. This shouldn't be modified
during export and import -->
        <xs:attribute name="resource_type" type="xs:integer" use="required"/>
        <xs:attribute name="type" type="xs:integer" use="optional"/>
        <xs:attribute name="update" type="xs:integer" use="required"/>
    </xs:complexType>

    <xs:complexType name="rolePermissionType" mixed="true">
        <xs:complexContent>
```

```
        <xs:extension base="rolePermissionAttrType">
            <xs:sequence>
                <xs:element name="permission" type="permissionAttrType"
                    minOccurs="1" maxOccurs="unbounded"/>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>


    <xs:complexType name="rolepermissionsType">
        <xs:sequence>
            <xs:element name="rolepermission" type="rolePermissionType" minOccurs="1"
                maxOccurs="unbounded"/>
            </xs:sequence>
        </xs:complexType>


        <xs:complexType name="AccessControlMappingFileStructureType">
            <xs:sequence>
                <xs:element name="users" type="usersType" minOccurs="0" maxOccurs="1"/>
                <xs:element name="rolepermissions" type="rolepermissionsType"
                    minOccurs="0"/>
                </xs:sequence>
            </xs:complexType>

            <xs:element name="accesscontrolmapping"
                type="AccessControlMappingFileStructureType"/>
```

importAccessControlMapping API

This operation imports the users, permissions, user-roles, role-permissions from the archive file to the target environment.

Input Parameters

importAccessControlMapping element contains the following:

- type: string
  - all is for users, permissions, user-roles, role-permissions
  - user is for users, user-roles
  - permission is for permissions, roles-permissions filename: string (Archive file name)
- filename: string (archive file name)
- location: string (location to which the archive file will be retrieved for import)

Output Parameters

accessControlMappingImportResponse element contains the following:

- status: string (success or fail)

Schema

```
<xs:complexType name="importAccessControlMappingType">
    <xs:complexType>
        <xs:sequence>
            <xs:element name="type">
                <xs:simpleType>
                    <xs:restriction base="xs:token">
```

```
                        <xs:enumeration value="all"/>
                        <xs:enumeration value="user"/>
                        <xs:enumeration value="permission"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:element>
            <xs:element name="filename"  />
            <xs:element name="location" />
        </xs:sequence>
    </xs:complexType>
</xs:complexType>

<xs:complexType name="accessControlMappingImportResponseType">
    <xs:complexType>
        <xs:sequence>
            <xs:element name="status">
                <xs:simpleType>
                    <xs:restriction base="xs:token">
                        <xs:enumeration value="success"/>
                        <xs:enumeration value="fail"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:element>
        </xs:sequence>
    </xs:complexType>
</xs:complexType>
```

Sample PMI Client

AMC version 2.4 includes a sample PMI client, located in **/opt/amc/admin/pmiclient**. The client consists of four files:

- PMIClient.java—The Java class. Modify this file to change testing parameters
- PMIHostNameVerifier—This is a helper class for PMIClient. It should not need modification
- cclient.sh—a shell script to compile PMIClient
- rclient.sh—A shell script to run a compiled PMIClient

Use the following workflow to test the PMI client:

1. Set up PMIClient.java to test the desired PMI command. Start with loginUser (see the PMI operation sections below).
2. Use cclient.sh to compile it.
3. Use rclient.sh to run it.
4. Look at the returned XML string to see if it worked or not.
5. Go to Step 1 and make changes to PMIClient.java until all variations of the PMI operations are tested.

Methods that test various PMI operations are invoked from main().

loginUser

- Change AMC_DEFAULT_PORT to be appropriate AMC Port if necessary.
- Change AMC_DEFAULT_HOST with the name or IP address of the AMC

- Change main so that none of the PMI command methods is called except for doPMIServiceLoginUser().

You should not have to change doPMIServiceLoginUser() method. Note that this method must be called for all PMI operations.

getProjectList

- Change main so that getProjectList() is called.
- Nothing to change in getProjectList() as getProjectList takes no parameters.

exportProject

- Change main so that callProjectExport() is called.
- Modify callProjectExport by changing “projectName”, “filename” and “location” variables.

importProject

- Change main so that callProjectImport() is called.
- Modify callProjectImport by changing “projectName”, “filename”, “location”, “mappingFile”, “mappingLocation” and “importType” variables.

getDRsList

- Change main so that getDRsList() is called.
- Modify getDRsList by changing “projectName” and “statusType” variables.

deployDR

- Change main so that deployDR() is called.
- Modify deployDR by changing “drid” variable.

callExportAccessControlMapping

- Change main so that callExportAccessControlMapping() is called.
- Modify callExportAccessControlMapping by changing “fileName”, “location”, and “type” variables.

callImportAccessControlMapping

- Change main so that callImportAccessControlMapping() is called.
- Modify callImportAccessControlMapping by changing “fileName”, “location”, and “type” variables.

Message Log Schemas

This section contains scripts that configure an Oracle or Sybase database for message log. For message log configuration instructions, see the “Message Log Domain” section on page 3-2.

Note Before running either of these scripts, be sure to remove any earlier versions of the message log schema.

Oracle Schema

Log in to SQLPlus as the user created for Message Log, then run this script to configure an Oracle database.

```
CREATE TABLE MESSAGE_LOG_INSTANCE
(   "USE_COUNT"    number(18,0),
    "ID"           varchar2(100),
    "DESCRIPTION"  varchar2(256),
    "VERSION"      varchar2(100)
);

INSERT INTO MESSAGE_LOG_INSTANCE (ID,USE_COUNT,VERSION,DESCRIPTION) VALUES
('AONS-MLOG-001', 0, '1.0', 'Database for storing AONS message logs');

CREATE TABLE MESSAGE_LOG
(
        "LOGID"        number(28,0) not null primary key,
    "HOSTNAME"                      varchar2(64),
    "SOURCE_NODE_ID"                number(10,0),
    "ENTRY_TIME"     timestamp,
    "CREATION_TIME"  timestamp not null,
    "MESSAGE_ID"     varchar2(100) not null,
    "SESSION_ID"     varchar2(100),
    "DESTINATION"    varchar2(256),
    "NEXT_HOP"       varchar2(256),
    "SOURCE" varchar2(256),
    "SENDING_NODE"   varchar2(256),
    "FLOW_ID" varchar2(100),
    "BLADELET_ID"    varchar2(32),
    "FLOW_NAME"      varchar2(100),
    "BLADELET_NAME"  varchar2(100),
    "CONTENT_TYPE"   varchar2(64),
    "PAYLOAD_TYPE"   varchar2(32),
    "MESSAGE_TYPE"   varchar2(32),
    "MESSAGE_CLASS"  varchar2(64),
    "PROTOCOL"       varchar2(32),
    "LOG_VERSION"    varchar2(10),
    "LOG_TYPE"       varchar2(32),
    "LOG_LEVEL"      number(5),
    "SOAP_OPERATION"  varchar2(256),
    "STATUS"          number(10),
        "REASON"          varchar2(100),
    "PROTOCOL_HEADER" raw(2000),
    "CUSTOM_STRING1"  varchar2(32),
        "CUSTOM_STRING2"  varchar2(64),
        "CUSTOM_STRING3"  varchar2(128),
        "CUSTOM_STRING4"  varchar2(256),
        "CUSTOM_STRING5"  varchar2(1024),
    "CUSTOM_NUMBER1"  number(5,0),
    "CUSTOM_NUMBER2"  number(10,0),
    "CUSTOM_NUMBER3"  number(18,2)
);

CREATE INDEX "MESSAGE_ID_IDX" ON "MESSAGE_LOG" ("MESSAGE_ID");
CREATE INDEX "MESSAGE_DESTINATION_IDX" ON "MESSAGE_LOG" ("DESTINATION");
CREATE INDEX "MESSAGE_SOURCE_IDX" ON "MESSAGE_LOG" ("SOURCE");

CREATE TABLE SOURCE_NODE
(
    ID NUMBER(10) NOT NULL PRIMARY KEY,
    DN VARCHAR2(256) NOT NULL,
    CREATED_TIME NUMBER(18) NOT NULL,
```

```
    NODE_ID NUMBER(10),
    AMC_IP VARCHAR2(64),
    AMC_HOST_NAME VARCHAR2(256),
    AMC_ID VARCHAR2(256),
    AMC_VERSION VARCHAR2(64),
    VIRTUAL_NODE_ID NUMBER(10),
    VIRTUAL_NODE_NAME VARCHAR2(64)
);

create table MESSAGE_CONTENTS (
    LOGID  number(28,0) not null,
    CONTENT_TYPE    varchar2(64),
    NAME   varchar2(64),
    CONTENT  long raw,
    EXPRESSION varchar2(256),
        CONTENT_LENGTH  number(10,0)
);

CREATE TABLE FLOW_VARIABLES
(
    "LOGID" number(28,0) NOT NULL,
    "NAME"  varchar2(100),
    "VALUE" long raw,
    "TYPE"  varchar2(100)
);

CREATE SEQUENCE LOGID_SEQ
START WITH 1
INCREMENT BY 50000
NOMAXVALUE;

CREATE OR REPLACE PROCEDURE GET_LOGID_BLOCK  (
  blockSize out int,
  beginValue out number
 )
as
begin
  select LOGID_SEQ.nextval INTO beginValue from dual;
  select INCREMENT_BY INTO blockSize from USER_SEQUENCES;
  beginValue := beginValue - blockSize;
  return;
end;
/

select LOGID_SEQ.nextval from dual;
```

Sybase Schema

Log in to SQLAdvantage as the user created for Message Log, then run this script to configure a Sybase database.

```
CREATE TABLE MESSAGE_LOG_INSTANCE
(
    USE_COUNT    numeric(18),
    ID           varchar(100),
    DESCRIPTION  varchar(256),
    VERSION      varchar(100)
)

go
```

```
INSERT INTO MESSAGE_LOG_INSTANCE (ID,USE_COUNT,VERSION,DESCRIPTION) VALUES
('AONS-MLOG-001', 0, '1.0', 'Database for storing AONS message logs')

go


create table MESSAGE_LOG (
    LOGID                           numeric(28,0)                   not null primary key
,
    HOSTNAME                        varchar(64)                         null  ,
    SOURCE_NODE_ID                  numeric(10,0)                       null  ,
    ENTRY_TIME                      datetime                            null  ,
    CREATION_TIME                   datetime                        not null  ,
    MESSAGE_ID                      varchar(100)                    not null  ,
    SESSION_ID                      varchar(100)                        null  ,
    DESTINATION                     varchar(256)                        null  ,
    NEXT_HOP                        varchar(256)                        null  ,
    SOURCE                          varchar(256)                        null  ,
    SENDING_NODE                    varchar(256)                        null  ,
    FLOW_ID                         varchar(100)                        null  ,
    BLADELET_ID                     varchar(32)                         null  ,
    FLOW_NAME                       varchar(100)                        null  ,
    BLADELET_NAME                   varchar(100)                        null  ,
    CONTENT_TYPE                    varchar(64)                         null  ,
    PAYLOAD_TYPE                    varchar(32)                         null  ,
    MESSAGE_TYPE                    varchar(32)                         null  ,
    MESSAGE_CLASS                   varchar(64)                         null  ,
    PROTOCOL                        varchar(32)                         null  ,
    LOG_VERSION                     varchar(10)                         null  ,
    LOG_TYPE                        varchar(32)                         null  ,
    LOG_LEVEL                       numeric(5,0)                        null  ,
    SOAP_OPERATION                  varchar(256)                        null  ,
    STATUS                          int                                 null  ,
        REASON                          varchar(100)                        null  ,
    PROTOCOL_HEADER                 varbinary(2000)                     null  ,
    CUSTOM_STRING1  varchar(32)     null  ,
        CUSTOM_STRING2  varchar(64)                         null  ,
        CUSTOM_STRING3  varchar(128)                        null  ,
        CUSTOM_STRING4  varchar(256)                        null  ,
        CUSTOM_STRING5  varchar(1024)                       null  ,
    CUSTOM_NUMBER1  numeric(5,0)                        null  ,
    CUSTOM_NUMBER2  numeric(10,0)                       null  ,
    CUSTOM_NUMBER3  numeric(18,2)                       null
)

go


CREATE INDEX MESSAGE_ID_IDX ON MESSAGE_LOG (MESSAGE_ID)

go

CREATE INDEX MESSAGE_DESTINATION_IDX ON MESSAGE_LOG (DESTINATION)

go

CREATE INDEX MESSAGE_SOURCE_IDX ON MESSAGE_LOG (SOURCE)

go

CREATE TABLE SOURCE_NODE
(
    ID numeric(10,0) not null primary key,
    DN varchar(256) not null,
```

```
    CREATED_TIME numeric(18,0) not null,
    NODE_ID numeric(10,0) null,
    AMC_IP varchar(64) null,
    AMC_HOST_NAME varchar(256) null,
    AMC_ID varchar(256) null,
    AMC_VERSION varchar(64) null,
    VIRTUAL_NODE_ID numeric(10,0) null,
    VIRTUAL_NODE_NAME varchar(64) null
)

go

create table MESSAGE_CONTENTS (
    LOGID  numeric(28,0) not null,
    CONTENT_TYPE    varchar(64) null,
    NAME   varchar(64) null,
    CONTENT  image null,
    EXPRESSION varchar(256) null,
        CONTENT_LENGTH  int null
)

go


create table FLOW_VARIABLES (
    LOGID       numeric(28,0)   not null  ,
    NAME        varchar(100)    null  ,
    TYPE        varchar(100)    null  ,
    VALUE       image           null
)

go


create table LOGID_KEY (

    ID      numeric(28,0)  not null
)

go

insert into LOGID_KEY values (1)

go

CREATE PROCEDURE GET_LOGID_BLOCK
@blockSize int output,
@beginValue numeric(28,0) output
AS
  BEGIN
    select @beginValue=ID from LOGID_KEY
    select @blockSize=50000
    update LOGID_KEY set ID=@beginValue + @blockSize
 END


go


sp_procxmode 'GET_LOGID_BLOCK', chained
go
```