The Application Level Events (ALE) Specification Version 1.1.1 Part II: XML and SOAP Bindings

EPCglobal Ratified Standard

13 March 2009

Previous Version: 1.1

Abstract

This document specifies an interface through which clients may interact with filtered, consolidated EPC data and related data from a variety of sources. The design of this interface recognizes that in most EPC processing systems, there is a level of processing that reduces the volume of data that comes directly from EPC data sources such as RFID readers into coarser "events" of interest to applications. It also recognizes that decoupling these applications from the physical layers of infrastructure offers cost and flexibility advantages to technology providers and end-users alike. The interface described herein, and the functionality it implies, is called "Application Level Events," or ALE.

The role of the ALE interface within the EPCglobal Network Architecture is to provide independence between the infrastructure components that acquire the raw EPC data, the architectural component(s) that filter & count that data, and the applications that use the data. This allows changes in one without requiring changes in the other, offering significant benefits to both the technology provider and the end-user. The ALE interface described in the present specification achieves this independence through five means:

- It provides a means for clients to specify, in a high-level, declarative way, what data they are interested in or what operations they want performed, without dictating an implementation. The interface is designed to give implementations the widest possible latitude in selecting strategies for carrying out client requests; such strategies may be influenced by performance goals, the native abilities of readers or other devices which may carry out certain filtering or counting operations at the level of firmware or RF protocol, and so forth.

- It provides a standardized format for reporting accumulated, filtered data and results from carrying out operations that is largely independent of where the data originated or how it was processed.

- It abstracts the channels through which data carriers are accessed into a higher-level notion of "logical reader," often synonymous with "location," hiding from clients the details of exactly what physical devices were used to interact with data relevant to a particular logical location. This allows changes to occur at the physical layer (for example, replacing a 2-port multi-antenna reader at a loading dock door with three "smart antenna" readers) without affecting client applications. Similarly, it abstracts away the fine-grained details of how data is gathered (*e.g.*, how many individual tag read attempts were carried out). These features of abstraction are a consequence of the way the data specification and reporting aspects of the interface are designed.

- It abstracts the addressing of information stored on Tags and other data carriers into a higher-level notion of named, typed "fields," hiding from clients the details of how a particular data element is encoded into a bit-level representation and stored at a particular address within a data carrier's memory. This allows application logic to remain invariant despite differences between the memory organization of different data carriers (for example, differences between Gen 1 and Gen 2 RFID Tags), and

also shields application logic from having to understand complex layout or data parsing rules.

- It provides a security mechanism so that administrators may choose which operations a given application may perform, as a policy that is decoupled from application logic itself.

Part I [ALE1.1 Part1] specifies at an abstract level all interfaces that are part of the ALE specification, using UML notation. This Part II specifies XML-based wire protocol bindings of the interfaces, including XSD schemas for all data types, WS-I compliant WSDL definitions of SOAP bindings of the service interfaces, and several XML-based bindings of callback interfaces used in certain modes of reading and writing data. Implementations may provide additional bindings of the API, including bindings to particular programming languages.

Audience for this document

The target audience for this specification includes:

- EPC Middleware vendors
- Reader vendors
- Application developers
- System integrators

Status of this document

This section describes the status of this document at the time of its publication. Other documents may supersede this document. The latest status of this document series is maintained at EPCglobal. See `www.epcglobalinc.org` for more information.

The Technical Steering Committee (TSC) approved the errata fixes to ALE 1.1, Part 1, that was originally ratified on February 27, 2008. Part 2's title page was updated to carry the same version number, 1.1.1, as Part 1, on March 13, 2009.

Comments on this document should be sent to the EPCglobal Software Action Group Filtering and Collection Working Group mailing list `sag_fc1_1_wg@lists.epcglobalinc.org`.

Errata Fixed in ALE 1.1.1

All fixed errata in ALE 1.1.1 apply to Part I of the specification; there are no changes to Part II. For a comparison between ALE 1.1 and ALE 1.0, please see Section 16 of Part I.

Table of Contents

1 Introduction

This document is Part II of the ALE 1.1 specification. This Part II specifies bindings of the classes and interfaces specified in Part I [ALE1.1Part1] that are based on XML [XML1.0]. In particular, this document specifies XML schemas for all classes defined in Part I, WSDL specifications of WS-I compliant SOAP interfaces of all interfaces defined in Part I, and XML-based bindings for the callback interfaces of the Reading and Writing APIs.

Implementations may provide additional bindings of the API, including bindings to particular programming languages.

2 Terminology and Typographical Conventions

Within this specification, the terms SHALL, SHALL NOT, SHOULD, SHOULD NOT, MAY, NEED NOT, CAN, and CANNOT are to be interpreted as specified in Annex G of the ISO/IEC Directives, Part 2, 2001, 4th edition [ISODir2]. When used in this way, these terms will always be shown in ALL CAPS; when these words appear in ordinary typeface they are intended to have their ordinary English meaning.

All sections of this document, with the exception of Section 1, are normative, except where explicitly noted as non-normative.

The following typographical conventions are used throughout the document:

- ALL CAPS type is used for the special terms from [ISODir2] enumerated above.
- `Monospace` type is used to denote programming language, UML, and XML identifiers, as well as for the text of XML documents.

➢ Placeholders for changes that need to be made to this document prior to its reaching the final stage of approved EPCglobal specification are prefixed by a rightward-facing arrowhead, as this paragraph is.

3 XML Schemas

This section defines the standard XML representation for all data types used by the five ALE APIs, including `ECSpec` instances and `ECReports` instances of the Reading API, `CCSpec` instances and `CCReports` instances of the Writing API, and other types used by the Writing API, the Tag Memory API, the Logical Reader API, and the Access Control API. All schemas are specified using the W3C XML Schema language [XSD1, XSD2]. Samples are also shown.

The schemas defined herein conform to EPCglobal standard schema design rules. The schema below imports the EPCglobal standard base schema, as mandated by the design rules.

3.1 Organization of the XML Schemas

Because an implementation may not implement all five ALE APIs, the schemas are divided into separate files, one per API. In addition, types shared by the Reading API and Writing API are defined in a separate file that is shared by the main file for the Reading and Writing APIs, and the five main schema files import the standard EPCglobal schema file.

All schemas are defined in the same XML namespace, which is `urn:epcglobal:ale:xsd:1`.

The seven schema files are summarized in the table below:

Schema	Section	Imported Schemas	Top-level Elements
`EPCglobal`	3.3	(none)	(none)
`ale-common`	3.4	`EPCglobal`	(none)
`ale`	3.5	`EPCglobal` `ale-common`	`ECSpec` `ECReports`

`alecc`	3.6	`EPCglobal`
		`ale-common`
			`CCSpec`
			`CCParameterList`
			`CCReports`
			`EPCCacheSpec`
			`EPCPatternList`
			`AssocTableSpec`
			`AssocTableEntryList`
			`RNGSpec`

`aletm`	3.7	`EPCglobal`	`TMSpec`
`alelr`	3.8	`EPCglobal`	`LRSpec` `LRProperty`

`aleac`	3.9	`EPCglobal`	`ACPermission`
			`ACRole`
			`ACClientIdentity`

3.2 Extensibility Mechanism

The XML schema in this section implements the `<<extension point>>` given in the UML using a methodology described in [XMLVersioning]. This methodology provides for both vendor extension, and for extension by EPCglobal in future versions of this specification or in supplemental specifications. Extensions introduced through this mechanism will be *backward compatible*, in that documents conforming to older versions of the schema will also conform to newer versions of the standard schema and to schema containing vendor-specific extensions. Extensions will also be *forward compatible*, in that documents that contain vendor extensions or that conform to newer versions of the standard schema will also conform to older versions of the schema.

When a document contains extensions (vendor-specific or standardized in newer versions of schema), it may conform to more than one schema. For example, a document containing vendor extensions to the ALE 1.1 schema will conform both to the EPCglobal ALE 1.1 schema and to a vendor-specific schema that includes the vendor extensions. In this example, when the document is parsed using the standard schema there will be no validation of the extension elements and attributes, but when the document is parsed using the vendor-specific schema the extensions will be validated. Similarly, a document containing new features introduced in the ALE 1.1 schema will conform both to the ALE 1.0 schema and to the ALE 1.1 schema, but validation of the new features will only be available using the ALE 1.1 schema.

The UML for ALE consists of three kinds of definitions: classes, interfaces, and enumerations. Extensibility for each of these is provided via a different XSD-level mechanism, as defined below.

3.2.1 Extensibility Design Rules – Classes

The design rules for the extensibility pattern used for classes are based on [XMLVersioning]. The design rules are as follows:

- For each UML class in which `<<extension point>>` occurs, include an `xsd:anyAttribute` declaration in the corresponding XSD type. This declaration provides for the addition of new attributes, either in subsequent versions of the standard schema or in vendor-specific schema.

- For each UML class in which `<<extension point>>` occurs, include an optional (`minOccurs` = 0) element named `extension` in the corresponding XSD type. The type declared for the `extension` element will always be as follows:

```
<xsd:sequence>
    <xsd:any processContents="lax" minOccurs="1" maxOccurs="unbounded"
             namespace="##local"/>
</xsd:sequence>
<xsd:anyAttribute processContents="lax"/>
```

  This declaration provides for forward-compatibility with new elements introduced into subsequent versions of the standard schema.

- For each UML class in which `<<extension point>>` occurs, include at the end of the element list within the corresponding XSD type a declaration

```
    <xsd:any processContents="lax" minOccurs="0" maxOccurs="unbounded"
             namespace="##other"/>
```

  This declaration provides for forward-compatibility with new elements introduced in vendor-specific schema.

The rules for adding vendor-specific extensions to the schema are as follows:

- Vendor-specific attributes may be added to any XSD type corresponding to a UML class in which `<<extension point>>` occurs. Vendor-specific attributes SHALL NOT be in the EPCglobal ALE namespace (`urn:epcglobal:ale:xsd:1`). Vendor-specific attributes SHALL be in a namespace whose namespace URI has the vendor as the owning authority. (In schema parlance, this means that all vendor-specific attributes must have `qualified` as their form.) For example, the namespace URI may be an HTTP URL whose authority portion is a domain name owned by the vendor, a URN having a URN namespace identifier issued to the vendor by IANA, an OID URN whose initial path is a Private Enterprise Number assigned to the vendor, etc. Declarations of vendor-specific attributes SHALL specify `use="optional"`.

- Vendor-specific elements may be added to any XSD type corresponding to a UML class in which `<<extension point>>` occurs. Vendor-specific elements SHALL NOT be in the EPCglobal ALE namespace (`urn:epcglobal:ale:xsd:1`). Vendor-specific elements SHALL be in a namespace whose namespace URI has the vendor as the owning authority (as described above). (In schema parlance, this means that all vendor-specific elements must have `qualified` as their form.)

To create a schema that contains vendor extensions, replace the `<xsd:any … namespace="##other"/>` declaration with a content group reference to a group defined in the vendor namespace; e.g., `<xsd:group ref="vendor:VendorExtension">`. In the schema file defining elements for the vendor namespace, define a content group using a declaration of the following form:

```
 <xsd:group name="VendorExtension">
   <xsd:sequence>
     <!--
       Definitions or references to vendor elements
       go here.  Each SHALL specify minOccurs="0".
     -->
     <xsd:any processContents="lax"
              minOccurs="0" maxOccurs="unbounded"
              namespace="##other"/>
   </xsd:sequence>
</xsd:group>
```

(In the foregoing illustrations, `vendor` and `VendorExtension` may be any strings the vendor chooses.)

Explanation (non-normative): Because vendor-specific elements must be optional, including references to their definitions directly into the ALE schema would violate the XML Schema Unique Particle Attribution constraint, because the `<xsd:any …>` element in the ALE schema can also match vendor-specific elements. Moving the `<xsd:any …>` into the vendor's schema avoids this problem, because `##other` in that schema means "match an element that has a namespace other than the vendor's namespace." This does not conflict with standard elements, because the element form default for the standard ALE schema is `unqualified`, and hence the `##other` in the vendor's schema does not match standard ALE elements, either.

The rules for adding attributes or elements to future versions of the ALE standard schema are as follows:

- Standard attributes may be added to any XSD type corresponding to a UML class in which `<<extension point>>` occurs. Standard attributes SHALL NOT be in any namespace, and SHALL NOT conflict with any existing standard attribute name.
- Standard elements may be added to any XSD type corresponding to a UML class in which `<<extension point>>` occurs. New elements are added using the following rules:
  - Find the innermost `extension` element type.
  - Replace the `<xsd:any … namespace="##local"/>` declaration with (a) new elements (which SHALL NOT be in any namespace); followed by (b) a new `extension` element whose type is constructed as described before. In subsequent revisions of the standard schema, new standard elements will be added within this new `extension` element rather than within this one.

Explanation (non-normative): the reason that new standard attributes and elements are specified above not to be in any namespace is to be consistent with the ALE schema's attribute and element form default of `unqualified`*.*

As noted in Section 6.1, in the schema for the ALE Reading API there are already several instances of this pattern being applied to introduce new features in ALE 1.1.

3.2.2 Extensibility Design Rules – Interfaces

All five ALE APIs include an `<<extension point>>` in the UML for the respective interfaces. The extension point for interfaces is implemented simply by noting that it is possible to add additional methods to the WSDL without affecting the existing methods.

A vendor implementation MAY add additional methods to an ALE API, provided that the name of a vendor extension method SHALL NOT conflict with existing methods.

Future versions of the ALE specification may add additional methods to the APIs. For back-compatibility, no incompatible changes will be made to existing methods.

3.2.3 Extensibility Design Rules – Enumerations

Enumerated types having an `<<extension point>>` in their UML are made extensible in the following manner. In the schema, such types are declared as equivalent to `xsd:string`. This provides for forward compatibility, as any vendor extension value or value defined in a future version of the specification will be valid according to the schema. A comment in the schema specifies the values that are defined in this version of the specification.

An implementation MAY extend an enumeration by allowing additional values beyond those defined in the specification. Vendor extension values SHALL take the form of absolute URIs [URI], where the URI has the vendor as the owning authority. For example, the URI for a vendor extension enumeration value may be an HTTP URL whose authority portion is a domain name owned by the vendor, a URN having a URN namespace identifier issued to the vendor by IANA, an OID URN whose initial path is a Private Enterprise Number assigned to the vendor, etc.

Future versions of the ALE specification may add additional values to enumerations. Any enumeration value specified in this or future versions of the specification will be strings not containing a colon (`:`) character, and therefore will never clash with vendor extension values that take the form of absolute URIs.

3.3 EPCglobal Base Schema

The XML schemas for all ALE APIs make reference to the EPCglobal Base Schema. This schema is reproduced below.

```
<xsd:schema targetNamespace="urn:epcglobal:xsd:1"
            xmlns:epcglobal="urn:epcglobal:xsd:1"
            xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            elementFormDefault="unqualified"
            attributeFormDefault="unqualified"
            version="1.0">
  <xsd:annotation>
    <xsd:documentation>
      <epcglobal:copyright>Copyright (C) 2004 Epcglobal Inc., All Rights
Reserved.</epcglobal:copyright>
      <epcglobal:disclaimer>EPCglobal Inc., its members, officers, directors, employees,
or agents shall not be liable for any injury, loss, damages, financial or otherwise,
arising from, related to, or caused by the use of this document.  The use of said
document shall constitute your express consent to the foregoing
exculpation.</epcglobal:disclaimer>
      <epcglobal:specification>EPCglobal common components Version
1.0</epcglobal:specification>
    </xsd:documentation>
  </xsd:annotation>
  <xsd:complexType name="Document" abstract="true">
    <xsd:annotation>
      <xsd:documentation xml:lang="en">
         EPCglobal document properties for all messages.
      </xsd:documentation>
    </xsd:annotation>
    <xsd:attribute name="schemaVersion" type="xsd:decimal" use="required">
      <xsd:annotation>
        <xsd:documentation xml:lang="en">
           The version of the schema corresponding to which the instance conforms.
        </xsd:documentation>
      </xsd:annotation>
    </xsd:attribute>
    <xsd:attribute name="creationDate" type="xsd:dateTime" use="required">
      <xsd:annotation>
        <xsd:documentation xml:lang="en">
           The date the message was created. Used for auditing and logging.
        </xsd:documentation>
      </xsd:annotation>
    </xsd:attribute>
  </xsd:complexType>
  <xsd:complexType name="EPC">
    <xsd:annotation>
      <xsd:documentation xml:lang="en">
         EPC represents the Electronic Product Code.
      </xsd:documentation>
    </xsd:annotation>
    <xsd:simpleContent>
      <xsd:extension base="xsd:string"/>
    </xsd:simpleContent>
  </xsd:complexType>
</xsd:schema>
```

3.4 Schema for Datatypes Common to the Reading API and Writing API

The following is an XML Schema (XSD) defining common data types used in both the Reading API and the Writing API.

Explanation (non-normative): Because an implementation MAY support only the Reading API, only the Writing API, or both, it is more convenient to have the common data types in a separate schema file that is imported by the main schema files for the Reading and Writing APIs. In this way, an implementation supporting only the Reading API need only concern itself with the schema file in this section together with the main Reading API schema file (similarly for an implementation that supports only the Writing API), and an implementation that implements both the Reading API and the Writing API can use all three files with no duplication.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            xmlns:ale="urn:epcglobal:ale:xsd:1"
            targetNamespace="urn:epcglobal:ale:xsd:1" elementFormDefault="unqualified"
            attributeFormDefault="unqualified" version="1.0">
  <!-- COMMON ELEMENTS -->

  <!-- COMMON TYPES -->

  <xsd:complexType name="ECFieldSpec">
    <xsd:sequence>
      <xsd:element name="fieldname" type="xsd:string"/>
      <xsd:element name="datatype" type="xsd:string" minOccurs="0"/>
      <xsd:element name="format" type="xsd:string" minOccurs="0"/>
      <xsd:element name="extension" type="ale:ECFieldSpecExtension" minOccurs="0"/>
      <xsd:any processContents="lax" minOccurs="0" maxOccurs="unbounded"
               namespace="##other"/>
    </xsd:sequence>
    <xsd:anyAttribute processContents="lax"/>
  </xsd:complexType>

  <xsd:complexType name="ECFieldSpecExtension">
    <xsd:sequence>
      <xsd:any processContents="lax" minOccurs="1" maxOccurs="unbounded"
               namespace="##local"/>
    </xsd:sequence>
    <xsd:anyAttribute processContents="lax"/>
  </xsd:complexType>

  <xsd:complexType name="ECFilterListMember">
    <xsd:sequence>
      <xsd:element name="includeExclude" type="ale:ECIncludeExclude"/>
      <xsd:element name="fieldspec" type="ale:ECFieldSpec"/>
      <xsd:element name="patList">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element name="pat" type="xsd:string" maxOccurs="unbounded"/>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
      <xsd:element name="extension" type="ale:ECFilterListMemberExtension"
                   minOccurs="0"/>
      <xsd:any processContents="lax" minOccurs="0" maxOccurs="unbounded"
               namespace="##other"/>
    </xsd:sequence>
    <xsd:anyAttribute processContents="lax"/>
  </xsd:complexType>

  <xsd:complexType name="ECFilterListMemberExtension">
    <xsd:sequence>
```

```
      <xsd:any processContents="lax" minOccurs="1" maxOccurs="unbounded"
               namespace="##local"/>
    </xsd:sequence>
    <xsd:anyAttribute processContents="lax"/>
  </xsd:complexType>

  <!-- The ECIncludeExclude type is an enumerated type.
       The following strings are legal values for this type:
         INCLUDE
         EXCLUDE
        -->
  <xsd:simpleType name="ECIncludeExclude">
    <xsd:restriction base="xsd:string"/>
  </xsd:simpleType>

  <xsd:complexType name="ECReaderStat">
    <xsd:sequence>
      <xsd:element name="readerName" type="xsd:string"/>
      <xsd:element name="sightings" minOccurs="0">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element name="sighting" type="ale:ECSightingStat" minOccurs="0"
                         maxOccurs="unbounded"/>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
    </xsd:sequence>
  </xsd:complexType>

  <xsd:complexType name="ECSightingStat"/>

  <xsd:complexType name="ECTime">
    <xsd:simpleContent>
      <xsd:extension base="xsd:long">
        <xsd:attribute name="unit" type="ale:ECTimeUnit" use="required"/>
      </xsd:extension>
    </xsd:simpleContent>
  </xsd:complexType>

  <!-- The ECTimeUnit type is an enumerated type.
       The following strings are legal values for this type:
         MS
       An implementation may also recognize additional strings as extensions.
        -->
  <xsd:simpleType name="ECTimeUnit">
    <xsd:restriction base="xsd:string"/>
  </xsd:simpleType>

  <xsd:simpleType name="ECTrigger">
    <xsd:restriction base="xsd:string"/>
  </xsd:simpleType>
</xsd:schema>
```

3.5 ALE Reading API Schema

The following is an XML Schema (XSD) for the ALE Reading API, defining both `ECSpec` and `ECReports` as top level elements.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            xmlns:ale="urn:epcglobal:ale:xsd:1"
            targetNamespace="urn:epcglobal:ale:xsd:1"
            xmlns:epcglobal="urn:epcglobal:xsd:1" elementFormDefault="unqualified"
            attributeFormDefault="unqualified" version="1.0">
  <xsd:import namespace="urn:epcglobal:xsd:1" schemaLocation="EPCglobal.xsd"/>
  <xsd:include schemaLocation="EPCglobal-ale-1_1-common.xsd"/>
  <!-- ALE ELEMENTS -->
  <xsd:element name="ECSpec" type="ale:ECSpec"/>
```

```
<xsd:element name="ECReports" type="ale:ECReports"/>

<!-- ALE TYPES -->

<xsd:complexType name="ECBoundarySpec">
  <xsd:sequence>
    <xsd:element name="startTrigger" type="ale:ECTrigger" minOccurs="0"/>
    <xsd:element name="repeatPeriod" type="ale:ECTime" minOccurs="0"/>
    <xsd:element name="stopTrigger" type="ale:ECTrigger" minOccurs="0"/>
    <xsd:element name="duration" type="ale:ECTime" minOccurs="0"/>
    <xsd:element name="stableSetInterval" type="ale:ECTime" minOccurs="0"/>
    <xsd:element name="extension" type="ale:ECBoundarySpecExtension" minOccurs="0"/>
    <xsd:any processContents="lax" minOccurs="0" maxOccurs="unbounded"
             namespace="##other"/>
  </xsd:sequence>
  <xsd:anyAttribute processContents="lax"/>
</xsd:complexType>

<xsd:complexType name="ECBoundarySpecExtension">
  <xsd:sequence>
    <xsd:element name="startTriggerList" minOccurs="0">
      <xsd:complexType>
        <xsd:sequence>
          <xsd:element name="startTrigger" type="ale:ECTrigger" minOccurs="0"
                       maxOccurs="unbounded"/>
        </xsd:sequence>
      </xsd:complexType>
    </xsd:element>
    <xsd:element name="stopTriggerList" minOccurs="0">
      <xsd:complexType>
        <xsd:sequence>
          <xsd:element name="stopTrigger" type="ale:ECTrigger" minOccurs="0"
                       maxOccurs="unbounded"/>
        </xsd:sequence>
      </xsd:complexType>
    </xsd:element>
    <xsd:element name="whenDataAvailable" type="xsd:boolean" minOccurs="0"/>
    <xsd:element name="extension" type="ale:ECBoundarySpecExtension2" minOccurs="0"/>
  </xsd:sequence>
  <xsd:anyAttribute processContents="lax"/>
</xsd:complexType>

<xsd:complexType name="ECBoundarySpecExtension2">
  <xsd:sequence>
    <xsd:any processContents="lax" minOccurs="1" maxOccurs="unbounded"
             namespace="##local"/>
  </xsd:sequence>
  <xsd:anyAttribute processContents="lax"/>
</xsd:complexType>

<xsd:complexType name="ECFilterSpec">
  <xsd:sequence>
    <xsd:element name="includePatterns" minOccurs="0">
      <xsd:complexType>
        <xsd:sequence>
          <xsd:element name="includePattern" type="xsd:string" minOccurs="0"
                       maxOccurs="unbounded"/>
        </xsd:sequence>
      </xsd:complexType>
    </xsd:element>
    <xsd:element name="excludePatterns" minOccurs="0">
      <xsd:complexType>
        <xsd:sequence>
          <xsd:element name="excludePattern" type="xsd:string" minOccurs="0"
                       maxOccurs="unbounded"/>
        </xsd:sequence>
      </xsd:complexType>
    </xsd:element>
    <xsd:element name="extension" type="ale:ECFilterSpecExtension" minOccurs="0"/>
    <xsd:any processContents="lax" minOccurs="0" maxOccurs="unbounded"
```

```
              namespace="##other"/>
    </xsd:sequence>
    <xsd:anyAttribute processContents="lax"/>
  </xsd:complexType>

  <xsd:complexType name="ECFilterSpecExtension">
    <xsd:sequence>
      <xsd:element name="filterList" minOccurs="0">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element name="filter" type="ale:ECFilterListMember" minOccurs="0"
                         maxOccurs="unbounded"/>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
      <xsd:element name="extension" type="ale:ECFilterSpecExtension2" minOccurs="0"/>
    </xsd:sequence>
    <xsd:anyAttribute processContents="lax"/>
  </xsd:complexType>

  <xsd:complexType name="ECFilterSpecExtension2">
    <xsd:sequence>
      <xsd:any processContents="lax" minOccurs="1" maxOccurs="unbounded"
              namespace="##local"/>
    </xsd:sequence>
    <xsd:anyAttribute processContents="lax"/>
  </xsd:complexType>

  <xsd:complexType name="ECGroupSpec">
    <xsd:sequence>
      <xsd:element name="pattern" type="xsd:string" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element name="extension" type="ale:ECGroupSpecExtension" minOccurs="0"/>
      <xsd:any processContents="lax" minOccurs="0" maxOccurs="unbounded"
              namespace="##other"/>
    </xsd:sequence>
    <xsd:anyAttribute processContents="lax"/>
  </xsd:complexType>

  <xsd:complexType name="ECGroupSpecExtension">
    <xsd:sequence>
      <xsd:element name="fieldspec" type="ale:ECFieldSpec" minOccurs="0"/>
      <xsd:element name="extension" type="ale:ECGroupSpecExtension2" minOccurs="0"/>
    </xsd:sequence>
    <xsd:anyAttribute processContents="lax"/>
  </xsd:complexType>

  <xsd:complexType name="ECGroupSpecExtension2">
    <xsd:sequence>
      <xsd:any processContents="lax" minOccurs="1" maxOccurs="unbounded"
              namespace="##local"/>
    </xsd:sequence>
    <xsd:anyAttribute processContents="lax"/>
  </xsd:complexType>

  <!-- The ECInitiationCondition type is an enumerated type.
       The following strings are legal values for this type:
         TRIGGER
         REPEAT_PERIOD
         REQUESTED
         UNDEFINE
       An implementation may also recognize additional strings as extensions.
        -->
  <xsd:simpleType name="ECInitiationCondition">
    <xsd:restriction base="xsd:string"/>
  </xsd:simpleType>

  <xsd:complexType name="ECReport">
    <xsd:sequence>
      <xsd:element name="group" type="ale:ECReportGroup" minOccurs="0"
                   maxOccurs="unbounded"/>
      <xsd:element name="extension" type="ale:ECReportExtension" minOccurs="0"/>
      <xsd:any processContents="lax" minOccurs="0" maxOccurs="unbounded"
               namespace="##other"/>
    </xsd:sequence>
    <xsd:attribute name="reportName" type="xsd:string" use="required"/>
    <xsd:anyAttribute processContents="lax"/>
  </xsd:complexType>

  <xsd:complexType name="ECReportExtension">
    <xsd:sequence>
      <xsd:any processContents="lax" minOccurs="1" maxOccurs="unbounded"
               namespace="##local"/>
    </xsd:sequence>
    <xsd:anyAttribute processContents="lax"/>
  </xsd:complexType>

  <xsd:complexType name="ECReportGroup">
    <xsd:sequence>
      <xsd:element name="groupList" type="ale:ECReportGroupList" minOccurs="0"/>
      <xsd:element name="groupCount" type="ale:ECReportGroupCount" minOccurs="0"/>
      <xsd:element name="extension" type="ale:ECReportGroupExtension" minOccurs="0"/>
      <xsd:any processContents="lax" minOccurs="0" maxOccurs="unbounded"
               namespace="##other"/>
    </xsd:sequence>
    <!-- The groupName attribute SHALL be omitted to indicate the default group. -->
    <xsd:attribute name="groupName" type="xsd:string" use="optional"/>
    <xsd:anyAttribute processContents="lax"/>
  </xsd:complexType>

  <xsd:complexType name="ECReportGroupExtension">
    <xsd:sequence>
      <xsd:any processContents="lax" minOccurs="1" maxOccurs="unbounded"
               namespace="##local"/>
    </xsd:sequence>
    <xsd:anyAttribute processContents="lax"/>
  </xsd:complexType>

  <xsd:complexType name="ECReportGroupCount">
    <xsd:sequence>
      <xsd:element name="count" type="xsd:int"/>
      <xsd:element name="extension" type="ale:ECReportGroupCountExtension"
                   minOccurs="0"/>
      <xsd:any processContents="lax" minOccurs="0" maxOccurs="unbounded"
               namespace="##other"/>
    </xsd:sequence>
    <xsd:anyAttribute processContents="lax"/>
  </xsd:complexType>

  <xsd:complexType name="ECReportGroupCountExtension">
    <xsd:sequence>
      <xsd:any processContents="lax" minOccurs="1" maxOccurs="unbounded"
               namespace="##local"/>
    </xsd:sequence>
    <xsd:anyAttribute processContents="lax"/>
  </xsd:complexType>

  <xsd:complexType name="ECReportGroupList">
    <xsd:sequence>
      <xsd:element name="member" type="ale:ECReportGroupListMember" minOccurs="0"
                   maxOccurs="unbounded"/>
      <xsd:element name="extension" type="ale:ECReportGroupListExtension" minOccurs="0"/>
      <xsd:any processContents="lax" minOccurs="0" maxOccurs="unbounded"
               namespace="##other"/>
    </xsd:sequence>
    <xsd:anyAttribute processContents="lax"/>
  </xsd:complexType>

  <xsd:complexType name="ECReportGroupListExtension">
    <xsd:sequence>
      <xsd:any processContents="lax" minOccurs="1" maxOccurs="unbounded"
```

```
              namespace="##local"/>
    </xsd:sequence>
    <xsd:anyAttribute processContents="lax"/>
  </xsd:complexType>

  <xsd:complexType name="ECReportGroupListMember">
    <xsd:sequence>
      <!-- Each of the following four elements SHALL be omitted if null. -->
      <xsd:element name="epc" type="epcglobal:EPC" minOccurs="0"/>
      <xsd:element name="tag" type="epcglobal:EPC" minOccurs="0"/>
      <xsd:element name="rawHex" type="epcglobal:EPC" minOccurs="0"/>
      <xsd:element name="rawDecimal" type="epcglobal:EPC" minOccurs="0"/>
      <xsd:element name="extension" type="ale:ECReportGroupListMemberExtension"
                   minOccurs="0"/>
      <xsd:any processContents="lax" minOccurs="0" maxOccurs="unbounded"
               namespace="##other"/>
    </xsd:sequence>
    <xsd:anyAttribute processContents="lax"/>
  </xsd:complexType>

  <xsd:complexType name="ECReportGroupListMemberExtension">
    <xsd:sequence>
      <xsd:element name="fieldList" minOccurs="0">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element name="field" type="ale:ECReportMemberField" minOccurs="0"
                         maxOccurs="unbounded"/>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
      <xsd:element name="stats" minOccurs="0">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element name="stat" type="ale:ECTagStat" minOccurs="0"
                         maxOccurs="unbounded"/>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
      <xsd:element name="extension" type="ale:ECReportGroupListMemberExtension2"
                   minOccurs="0"/>
    </xsd:sequence>
    <xsd:anyAttribute processContents="lax"/>
  </xsd:complexType>

  <xsd:complexType name="ECReportGroupListMemberExtension2">
    <xsd:sequence>
      <xsd:any processContents="lax" minOccurs="1" maxOccurs="unbounded"
               namespace="##local"/>
    </xsd:sequence>
    <xsd:anyAttribute processContents="lax"/>
  </xsd:complexType>

  <xsd:complexType name="ECReportMemberField">
    <xsd:sequence>
      <xsd:element name="name" type="xsd:string"/>
      <xsd:element name="value" type="xsd:string" minOccurs="0"/>
      <xsd:element name="fieldspec" type="ale:ECFieldSpec" minOccurs="0"/>
      <xsd:element name="extension" type="ale:ECReportMemberFieldExtension"
                   minOccurs="0"/>
      <xsd:any processContents="lax" minOccurs="0" maxOccurs="unbounded"
               namespace="##other"/>
    </xsd:sequence>
    <xsd:anyAttribute processContents="lax"/>
  </xsd:complexType>

  <xsd:complexType name="ECReportMemberFieldExtension">
    <xsd:sequence>
      <xsd:any processContents="lax" minOccurs="1" maxOccurs="unbounded"
               namespace="##local"/>
    </xsd:sequence>
    <xsd:anyAttribute processContents="lax"/>
  </xsd:complexType>

  <xsd:complexType name="ECReportOutputFieldSpec">
    <xsd:sequence>
      <xsd:element name="fieldspec" type="ale:ECFieldSpec"/>
      <xsd:element name="name" type="xsd:string" minOccurs="0"/>
      <xsd:element name="includeFieldSpecInReport" type="xsd:boolean" minOccurs="0"/>
      <xsd:element name="extension" type="ale:ECReportOutputFieldSpecExtension"
                   minOccurs="0"/>
      <xsd:any processContents="lax" minOccurs="0" maxOccurs="unbounded"
               namespace="##other"/>
    </xsd:sequence>
    <xsd:anyAttribute processContents="lax"/>
  </xsd:complexType>

  <xsd:complexType name="ECReportOutputFieldSpecExtension">
    <xsd:sequence>
      <xsd:any processContents="lax" minOccurs="1" maxOccurs="unbounded"
               namespace="##local"/>
    </xsd:sequence>
    <xsd:anyAttribute processContents="lax"/>
  </xsd:complexType>

  <xsd:complexType name="ECReportOutputSpec">
    <xsd:sequence>
      <xsd:element name="extension" type="ale:ECReportOutputSpecExtension"
                   minOccurs="0"/>
      <xsd:any processContents="lax" minOccurs="0" maxOccurs="unbounded"
               namespace="##other"/>
    </xsd:sequence>
    <xsd:attribute name="includeEPC" type="xsd:boolean" default="false"/>
    <xsd:attribute name="includeTag" type="xsd:boolean" default="false"/>
    <xsd:attribute name="includeRawHex" type="xsd:boolean" default="false"/>
    <xsd:attribute name="includeRawDecimal" type="xsd:boolean" default="false"/>
    <xsd:attribute name="includeCount" type="xsd:boolean" default="false"/>
    <xsd:anyAttribute processContents="lax"/>
  </xsd:complexType>

  <xsd:complexType name="ECReportOutputSpecExtension">
    <xsd:sequence>
      <xsd:element name="fieldList" minOccurs="0">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element name="field" type="ale:ECReportOutputFieldSpec" minOccurs="0"
                         maxOccurs="unbounded"/>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
      <xsd:element name="extension" type="ale:ECReportOutputSpecExtension2"
                   minOccurs="0"/>
    </xsd:sequence>
    <xsd:anyAttribute processContents="lax"/>
  </xsd:complexType>

  <xsd:complexType name="ECReportOutputSpecExtension2">
    <xsd:sequence>
      <xsd:any processContents="lax" minOccurs="1" maxOccurs="unbounded"
               namespace="##local"/>
    </xsd:sequence>
    <xsd:anyAttribute processContents="lax"/>
  </xsd:complexType>

  <xsd:complexType name="ECReports">
    <xsd:complexContent>
      <xsd:extension base="epcglobal:Document">
        <xsd:sequence>
          <xsd:element name="reports">
            <xsd:complexType>
              <xsd:sequence>
```

```
                  <xsd:element name="report" type="ale:ECReport" minOccurs="0"
                               maxOccurs="unbounded"/>
                </xsd:sequence>
              </xsd:complexType>
            </xsd:element>
            <xsd:element name="extension" type="ale:ECReportsExtension" minOccurs="0"/>
            <xsd:element name="ECSpec" type="ale:ECSpec" minOccurs="0"/>
            <xsd:any processContents="lax" minOccurs="0" maxOccurs="unbounded"
                     namespace="##other"/>
          </xsd:sequence>
          <xsd:attribute name="specName" type="xsd:string" use="required"/>
          <xsd:attribute name="date" type="xsd:dateTime" use="required"/>
          <xsd:attribute name="ALEID" type="xsd:string" use="required"/>
          <xsd:attribute name="totalMilliseconds" type="xsd:long" use="required"/>
          <xsd:attribute name="initiationCondition" type="ale:ECInitiationCondition"
                         use="optional"/>
          <xsd:attribute name="initiationTrigger" type="ale:ECTrigger" use="optional"/>
          <xsd:attribute name="terminationCondition" type="ale:ECTerminationCondition"
                         use="required"/>
          <xsd:attribute name="terminationTrigger" type="ale:ECTrigger" use="optional"/>
          <xsd:attribute name="schemaURL" type="xsd:string" use="optional"/>
          <xsd:anyAttribute processContents="lax"/>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="ECReportsExtension">
      <xsd:sequence>
        <xsd:any processContents="lax" minOccurs="1" maxOccurs="unbounded"
                 namespace="##local"/>
      </xsd:sequence>
      <xsd:anyAttribute processContents="lax"/>
    </xsd:complexType>

    <!-- The ECReportSetEnum type is an enumerated type.
         The following strings are legal values for this type:
           CURRENT
           ADDITIONS
           DELETIONS
         An implementation may also recognize additional strings as extensions.
          -->
    <xsd:simpleType name="ECReportSetEnum">
      <xsd:restriction base="xsd:string"/>
    </xsd:simpleType>

    <xsd:complexType name="ECReportSetSpec">
      <xsd:attribute name="set" type="ale:ECReportSetEnum" use="required"/>
    </xsd:complexType>

    <xsd:complexType name="ECReportSpec">
      <xsd:sequence>
        <xsd:element name="reportSet" type="ale:ECReportSetSpec"/>
        <xsd:element name="filterSpec" type="ale:ECFilterSpec" minOccurs="0"/>
        <xsd:element name="groupSpec" type="ale:ECGroupSpec" minOccurs="0"/>
        <xsd:element name="output" type="ale:ECReportOutputSpec"/>
        <xsd:element name="extension" type="ale:ECReportSpecExtension" minOccurs="0"/>
        <xsd:any processContents="lax" minOccurs="0" maxOccurs="unbounded"
                 namespace="##other"/>
      </xsd:sequence>
      <xsd:attribute name="reportName" type="xsd:string" use="required"/>
      <xsd:attribute name="reportIfEmpty" type="xsd:boolean" default="false"/>
      <xsd:attribute name="reportOnlyOnChange" type="xsd:boolean" default="false"/>
      <xsd:anyAttribute processContents="lax"/>
    </xsd:complexType>

    <xsd:complexType name="ECReportSpecExtension">
      <xsd:sequence>
        <xsd:element name="statProfileNames" minOccurs="0">
          <xsd:complexType>
            <xsd:sequence>
            <xsd:element name="statProfileName" type="ale:ECStatProfileName"
                         minOccurs="0" maxOccurs="unbounded"/>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
      <xsd:element name="extension" type="ale:ECReportSpecExtension2" minOccurs="0"/>
    </xsd:sequence>
    <xsd:anyAttribute processContents="lax"/>
  </xsd:complexType>

  <xsd:complexType name="ECReportSpecExtension2">
    <xsd:sequence>
      <xsd:any processContents="lax" minOccurs="1" maxOccurs="unbounded"
               namespace="##local"/>
    </xsd:sequence>
    <xsd:anyAttribute processContents="lax"/>
  </xsd:complexType>

  <xsd:complexType name="ECSpec">
    <xsd:complexContent>
      <xsd:extension base="epcglobal:Document">
        <xsd:sequence>
          <xsd:element name="logicalReaders">
            <xsd:complexType>
              <xsd:sequence>
                <xsd:element name="logicalReader" type="xsd:string"
                             maxOccurs="unbounded"/>
              </xsd:sequence>
            </xsd:complexType>
          </xsd:element>
          <xsd:element name="boundarySpec" type="ale:ECBoundarySpec"/>
          <xsd:element name="reportSpecs">
            <xsd:complexType>
              <xsd:sequence>
                <xsd:element name="reportSpec" type="ale:ECReportSpec"
                             maxOccurs="unbounded"/>
              </xsd:sequence>
            </xsd:complexType>
          </xsd:element>
          <xsd:element name="extension" type="ale:ECSpecExtension" minOccurs="0"/>
          <xsd:any processContents="lax" minOccurs="0" maxOccurs="unbounded"
                   namespace="##other"/>
        </xsd:sequence>
        <xsd:attribute name="includeSpecInReports" type="xsd:boolean" default="false"/>
        <xsd:anyAttribute processContents="lax"/>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>

  <xsd:complexType name="ECSpecExtension">
    <xsd:sequence>
      <xsd:element name="primaryKeyFields" minOccurs="0">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element name="primaryKeyField" type="xsd:string" minOccurs="0"
                         maxOccurs="unbounded"/>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
      <xsd:element name="extension" type="ale:ECSpecExtension2" minOccurs="0"/>
    </xsd:sequence>
    <xsd:anyAttribute processContents="lax"/>
  </xsd:complexType>

  <xsd:complexType name="ECSpecExtension2">
    <xsd:sequence>
      <xsd:any processContents="lax" minOccurs="1" maxOccurs="unbounded"
               namespace="##local"/>
    </xsd:sequence>
    <xsd:anyAttribute processContents="lax"/>
  </xsd:complexType>

  <!-- The ECStatProfileName type is an enumerated type.
       The following strings are legal values for this type:
         TagTimestamps
       An implementation may also recognize additional strings as extensions.
        -->
  <xsd:simpleType name="ECStatProfileName">
    <xsd:restriction base="xsd:string"/>
  </xsd:simpleType>

  <xsd:complexType name="ECTagStat">
    <xsd:sequence>
      <xsd:element name="profile" type="ale:ECStatProfileName"/>
      <xsd:element name="statBlocks" minOccurs="0">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element name="statBlock" type="ale:ECReaderStat" minOccurs="0"
                         maxOccurs="unbounded"/>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
    </xsd:sequence>
  </xsd:complexType>

  <xsd:complexType name="ECTagTimestampStat">
    <xsd:complexContent>
      <xsd:extension base="ale:ECTagStat">
        <xsd:sequence>
          <xsd:element name="firstSightingTime" type="xsd:dateTime"/>
          <xsd:element name="lastSightingTime" type="xsd:dateTime"/>
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>

  <!-- The ECTerminationCondition type is an enumerated type.
       The following strings are legal values for this type:
         TRIGGER
         DURATION
         STABLE_SET
         DATA_AVAILABLE
         UNREQUEST
         UNDEFINE
       An implementation may also recognize additional strings as extensions.
        -->
  <xsd:simpleType name="ECTerminationCondition">
    <xsd:restriction base="xsd:string"/>
  </xsd:simpleType>
</xsd:schema>
```

3.5.1 ECSpec – Example 1 (non-normative)

Here is an example `ECSpec` rendered into XML [XML1.0]:

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

<ale:ECSpec xmlns:ale="urn:epcglobal:ale:xsd:1"
            xmlns:epcglobal="urn:epcglobal:xsd:1"
            xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            schemaVersion="1.0"
            creationDate="2003-08-06T10:54:06.444-05:00">
    <logicalReaders>
        <logicalReader>dock_1a</logicalReader>
        <logicalReader>dock_1b</logicalReader>
    </logicalReaders>
    <boundarySpec>
        <startTrigger>http://example.com/trigger1</startTrigger>
        <repeatPeriod unit="MS">20000</repeatPeriod>
        <stopTrigger>http://example.com/trigger2</stopTrigger>
        <duration unit="MS">3000</duration>
    </boundarySpec>
    <reportSpecs>
        <reportSpec reportName="report1">
            <reportSet set="CURRENT"/>
            <output includeTag="true"/>
        </reportSpec>
        <reportSpec reportName="report2">
            <reportSet set="ADDITIONS"/>
            <output includeCount="true"/>
        </reportSpec>
        <reportSpec reportName="report3">
            <reportSet set="DELETIONS"/>
            <groupSpec>
               <pattern>urn:epc:pat:sgtin-96:X.X.X.*</pattern>
            </groupSpec>
            <output includeCount="true"/>
        </reportSpec>
    </reportSpecs>
</ale:ECSpec>
```

3.5.2 ECSpec – Example 2 (non-normative)

Here is a second example `ECSpec` rendered into XML [XML1.0]:

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

<ale:ECSpec xmlns:ale="urn:epcglobal:ale:xsd:1"
            xmlns:epcglobal="urn:epcglobal:xsd:1"
            xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            schemaVersion="1.0"
            creationDate="2007-05-14T10:10:10+02:00">
    <logicalReaders>
        <logicalReader>dock_1a</logicalReader>
        <logicalReader>dock_1b</logicalReader>
    </logicalReaders>
    <boundarySpec>
        <repeatPeriod unit="MS">20000</repeatPeriod>
        <duration unit="MS">3000</duration>
        <extension>
            <startTriggerList>
                <startTrigger>http://example.com/trigger1</startTrigger>
            </startTriggerList>
            <stopTriggerList>
                <stopTrigger>http://example.com/trigger2</stopTrigger>
            </stopTriggerList>
        </extension>
    </boundarySpec>
    <reportSpecs>
        <reportSpec reportName="report1">
            <reportSet set="CURRENT"/>
            <filterSpec>
                <extension>
                    <filterList>
                        <filter>
                            <includeExclude>INCLUDE</includeExclude>
                            <fieldspec>
                                <fieldname>epc</fieldname>
                            </fieldspec>
                            <patList>
                                <pat>urn:epc:pat:gid-96:*.*.*</pat>
                            </patList>
                        </filter>
                    </filterList>
                </extension>
            </filterSpec>
            <output includeTag="true">
                <extension>
```

```
                    <fieldList>
                        <field>
                            <fieldspec>
                                <fieldname>LotCode</fieldname>
                            </fieldspec>
                        </field>
                    </fieldList>
                </extension>
            </output>
        </reportSpec>
        <reportSpec reportName="report2">
            <reportSet set="ADDITIONS"/>
            <output includeCount="true"/>
        </reportSpec>
        <reportSpec reportName="report3">
            <reportSet set="DELETIONS"/>
            <groupSpec>
               <pattern>urn:epc:pat:sgtin-96:X.X.X.*</pattern>
            </groupSpec>
            <output includeCount="true"/>
        </reportSpec>
    </reportSpecs>
</ale:ECSpec>
```

3.5.3 ECReports – Example 1 (non-normative)

Here is an example `ECReports` rendered into XML [XML1.0]: This `ECReports` instance corresponds to the `ECSpec` example in Section 3.5.1, assuming that the ECSpec was defined using the name “embarcadère”.

```
<?xml version="1.0" encoding="UTF-8"?>
<ale:ECReports xmlns:ale="urn:epcglobal:ale:xsd:1"
               xmlns:epcglobal="urn:epcglobal:xsd:1"
               xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
               schemaVersion="1.0"
               creationDate="2003-08-06T10:54:06.444-05:00"
               specName="embarcad&#232;re"
               date="2003-08-06T10:54:06.444-05:00"
               ALEID="Edge34"
               totalMilliseconds="3034"
               initiationCondition="TRIGGER"
               initiationTrigger="http://example.com/trigger1"
               terminationCondition="DURATION">
    <reports>
        <report reportName="report1">
            <group>
                <groupList>
                    <member><tag>urn:epc:tag:gid-96:10.50.1000</tag></member>
                    <member><tag>urn:epc:tag:gid-96:10.50.1001</tag></member>
                </groupList>
            </group>
        </report>
        <report reportName="report2">
            <group><groupCount><count>6847</count></groupCount></group>
        </report>
        <report reportName="report3">
            <group name="urn:epc:pat:sgtin-96:3.0037000.012345.*">
               <groupCount><count>2</count></groupCount>
            </group>
            <group name="urn:epc:pat:sgtin-96:3.0037000.055555.*">
               <groupCount><count>3</count></groupCount>
            </group>
            <group>
               <groupCount><count>6842</count></groupCount>
            </group>
        </report>
    </reports>
</ale:ECReports>
```

3.5.4 ECReports – Example 2 (non-normative)

Here is an example `ECReports` rendered into XML [XML1.0]: This `ECReports` instance corresponds to the `ECSpec` example in Section 3.5.2, assuming the ECSpec was defined using the name “Spec1”.

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

<ale:ECReports xmlns:ale="urn:epcglobal:ale:xsd:1"
               xmlns:epcglobal="urn:epcglobal:xsd:1"
               xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
               schemaVersion="1.0"
               creationDate="2007-05-14T11:11:11+02:00"
               specName="Spec1"
               date="2007-05-14T11:11:11+02:00"
               ALEID="Edge34"
               totalMilliseconds="3034"
               initiationCondition="TRIGGER"
               initiationTrigger="http://example.com/trigger1"
               terminationCondition="TRIGGER"
               terminationTrigger="http://example.com/trigger2">
    <reports>
        <report reportName="report1">
            <group>
                <groupList>
                    <member>
                        <tag>urn:epc:tag:gid-96:10.50.1000</tag>
                        <extension>
                            <fieldList>
                                <field>
                                    <name>LotCode</name>
                                    <value>12345</value>
                                </field>
                            </fieldList>
                        </extension>
                    </member>
                    <member>
                        <tag>urn:epc:tag:gid-96:10.50.1001</tag>
                        <extension>
                            <fieldList>
                                <field>
                                    <name>LotCode</name>
                                    <value>12345</value>
                                </field>
                            </fieldList>
                        </extension>
                    </member>
                </groupList>
            </group>
        </report>
        <report reportName="report2">
            <group><groupCount><count>6847</count></groupCount></group>
        </report>
        <report reportName="report3">
            <group name="urn:epc:pat:sgtin-96:3.0037000.012345.*">
               <groupCount><count>2</count></groupCount>
            </group>
            <group name="urn:epc:pat:sgtin-96:3.0037000.055555.*">
               <groupCount><count>3</count></groupCount>
            </group>
            <group>
               <groupCount><count>6842</count></groupCount>
            </group>
        </report>
    </reports>
</ale:ECReports>
```

3.6 ALE Writing API Schema

The following is an XML Schema (XSD) for the ALE Writing API, defining `CCSpec`, `CCParameterList`, `CCReports`, `EPCCacheSpec`, `EPCPatternList`, `AssocTableSpec`, `AssocTableEntryList`, and `RNGSpec` as top level elements.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            xmlns:ale="urn:epcglobal:ale:xsd:1"
            targetNamespace="urn:epcglobal:ale:xsd:1"
            xmlns:epcglobal="urn:epcglobal:xsd:1" elementFormDefault="unqualified"
            attributeFormDefault="unqualified" version="1.0">
  <xsd:import namespace="urn:epcglobal:xsd:1" schemaLocation="EPCglobal.xsd"/>
  <xsd:include schemaLocation="EPCglobal-ale-1_1-common.xsd"/>
  <!-- ALECC ELEMENTS -->
  <xsd:element name="CCSpec" type="ale:CCSpec"/>
  <xsd:element name="CCReports" type="ale:CCReports"/>
  <xsd:element name="EPCCacheSpec" type="ale:EPCCacheSpec"/>
  <xsd:element name="EPCPatternList" type="ale:EPCPatternList"/>
  <xsd:element name="AssocTableSpec" type="ale:AssocTableSpec"/>
  <xsd:element name="AssocTableEntryList" type="ale:AssocTableEntryList"/>
  <xsd:element name="RNGSpec" type="ale:RNGSpec"/>

  <!-- ALECC TYPES -->

  <xsd:complexType name="AssocTableEntry">
    <xsd:sequence>
      <xsd:element name="key" type="xsd:string"/>
      <xsd:element name="value" type="xsd:string"/>
    </xsd:sequence>
  </xsd:complexType>

  <xsd:complexType name="AssocTableEntryList">
    <xsd:sequence>
      <xsd:element name="entries" minOccurs="0">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element name="entry" type="ale:AssocTableEntry" minOccurs="0"
                         maxOccurs="unbounded"/>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
    </xsd:sequence>
  </xsd:complexType>

  <xsd:complexType name="AssocTableSpec">
    <xsd:complexContent>
      <xsd:extension base="epcglobal:Document">
        <xsd:sequence>
          <xsd:element name="datatype" type="xsd:string"/>
          <xsd:element name="format" type="xsd:string"/>
          <xsd:element name="extension" type="ale:AssocTableSpecExtension"
                       minOccurs="0"/>
          <xsd:any processContents="lax" minOccurs="0" maxOccurs="unbounded"
                   namespace="##other"/>
        </xsd:sequence>
        <xsd:anyAttribute processContents="lax"/>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>

  <xsd:complexType name="AssocTableSpecExtension">
    <xsd:sequence>
      <xsd:any processContents="lax" minOccurs="1" maxOccurs="unbounded"
               namespace="##local"/>
    </xsd:sequence>
    <xsd:anyAttribute processContents="lax"/>
  </xsd:complexType>

```

```
<xsd:complexType name="CCBoundarySpec">
  <xsd:sequence>
    <xsd:element name="startTriggerList" minOccurs="0">
      <xsd:complexType>
        <xsd:sequence>
          <xsd:element name="startTrigger" type="ale:ECTrigger" minOccurs="0"
                       maxOccurs="unbounded"/>
        </xsd:sequence>
      </xsd:complexType>
    </xsd:element>
    <xsd:element name="repeatPeriod" type="ale:ECTime" minOccurs="0"/>
    <xsd:element name="stopTriggerList" minOccurs="0">
      <xsd:complexType>
        <xsd:sequence>
          <xsd:element name="stopTrigger" type="ale:ECTrigger" minOccurs="0"
                       maxOccurs="unbounded"/>
        </xsd:sequence>
      </xsd:complexType>
    </xsd:element>
    <xsd:element name="duration" type="ale:ECTime" minOccurs="0"/>
    <xsd:element name="noNewTagsInterval" type="ale:ECTime" minOccurs="0"/>
    <xsd:element name="tagsProcessedCount" type="xsd:int" minOccurs="0"/>
    <xsd:element name="afterError" type="xsd:boolean" minOccurs="0"/>
    <xsd:element name="extension" type="ale:CCBoundarySpecExtension" minOccurs="0"/>
    <xsd:any processContents="lax" minOccurs="0" maxOccurs="unbounded"
             namespace="##other"/>
  </xsd:sequence>
  <xsd:anyAttribute processContents="lax"/>
</xsd:complexType>

<xsd:complexType name="CCBoundarySpecExtension">
  <xsd:sequence>
    <xsd:any processContents="lax" minOccurs="1" maxOccurs="unbounded"
             namespace="##local"/>
  </xsd:sequence>
  <xsd:anyAttribute processContents="lax"/>
</xsd:complexType>

<xsd:complexType name="CCCmdReport">
  <xsd:sequence>
    <xsd:element name="tagReports" minOccurs="0">
      <xsd:complexType>
        <xsd:sequence>
          <xsd:element name="tagReport" type="ale:CCTagReport" minOccurs="0"
                       maxOccurs="unbounded"/>
        </xsd:sequence>
      </xsd:complexType>
    </xsd:element>
    <xsd:element name="extension" type="ale:CCCmdReportExtension" minOccurs="0"/>
    <xsd:any processContents="lax" minOccurs="0" maxOccurs="unbounded"
             namespace="##other"/>
  </xsd:sequence>
  <xsd:attribute name="cmdSpecName" type="xsd:string" use="required"/>
  <xsd:anyAttribute processContents="lax"/>
</xsd:complexType>

<xsd:complexType name="CCCmdReportExtension">
  <xsd:sequence>
    <xsd:any processContents="lax" minOccurs="1" maxOccurs="unbounded"
             namespace="##local"/>
  </xsd:sequence>
  <xsd:anyAttribute processContents="lax"/>
</xsd:complexType>

<xsd:complexType name="CCCmdSpec">
  <xsd:sequence>
    <xsd:element name="filterSpec" type="ale:CCFilterSpec" minOccurs="0"/>
    <xsd:element name="opSpecs" minOccurs="0">
      <xsd:complexType>
        <xsd:sequence>
```

```
            <xsd:element name="opSpec" type="ale:CCOpSpec" minOccurs="0"
                         maxOccurs="unbounded"/>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
      <xsd:element name="statProfileNames" minOccurs="0">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element name="statProfileName" type="ale:CCStatProfileName"
                         minOccurs="0" maxOccurs="unbounded"/>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
      <xsd:element name="extension" type="ale:CCCmdSpecExtension" minOccurs="0"/>
      <xsd:any processContents="lax" minOccurs="0" maxOccurs="unbounded"
               namespace="##other"/>
    </xsd:sequence>
    <xsd:attribute name="name" type="xsd:string" use="required"/>
    <xsd:attribute name="reportIfEmpty" type="xsd:boolean" default="false"/>
    <xsd:anyAttribute processContents="lax"/>
  </xsd:complexType>

  <xsd:complexType name="CCCmdSpecExtension">
    <xsd:sequence>
      <xsd:any processContents="lax" minOccurs="1" maxOccurs="unbounded"
               namespace="##local"/>
    </xsd:sequence>
    <xsd:anyAttribute processContents="lax"/>
  </xsd:complexType>

  <xsd:complexType name="CCFilterSpec">
    <xsd:sequence>
      <xsd:element name="filterList" minOccurs="0">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element name="filter" type="ale:ECFilterListMember" minOccurs="0"
                         maxOccurs="unbounded"/>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
      <xsd:element name="extension" type="ale:CCFilterSpecExtension" minOccurs="0"/>
      <xsd:any processContents="lax" minOccurs="0" maxOccurs="unbounded"
               namespace="##other"/>
    </xsd:sequence>
    <xsd:anyAttribute processContents="lax"/>
  </xsd:complexType>

  <xsd:complexType name="CCFilterSpecExtension">
    <xsd:sequence>
      <xsd:any processContents="lax" minOccurs="1" maxOccurs="unbounded"
               namespace="##local"/>
    </xsd:sequence>
    <xsd:anyAttribute processContents="lax"/>
  </xsd:complexType>

  <!-- The CCInitiationCondition type is an enumerated type.
       The following strings are legal values for this type:
         TRIGGER
         REPEAT_PERIOD
         REQUESTED
         UNDEFINE
       An implementation may also recognize additional strings as extensions.
        -->
  <xsd:simpleType name="CCInitiationCondition">
    <xsd:restriction base="xsd:string"/>
  </xsd:simpleType>

  <!-- The CCLockOperation type is an enumerated type.
       The following strings are legal values for this type:
         UNLOCK
        PERMAUNLOCK
        LOCK
        PERMALOCK
      An implementation may also recognize additional strings as extensions.
       -->
  <xsd:simpleType name="CCLockOperation">
    <xsd:restriction base="xsd:string"/>
  </xsd:simpleType>

  <xsd:complexType name="CCOpDataSpec">
    <xsd:sequence>
      <xsd:element name="specType" type="ale:CCOpDataSpecType"/>
      <xsd:element name="data" type="xsd:string"/>
      <xsd:element name="extension" type="ale:CCOpDataSpecExtension" minOccurs="0"/>
      <xsd:any processContents="lax" minOccurs="0" maxOccurs="unbounded"
               namespace="##other"/>
    </xsd:sequence>
    <xsd:anyAttribute processContents="lax"/>
  </xsd:complexType>

  <xsd:complexType name="CCOpDataSpecExtension">
    <xsd:sequence>
      <xsd:any processContents="lax" minOccurs="1" maxOccurs="unbounded"
               namespace="##local"/>
    </xsd:sequence>
    <xsd:anyAttribute processContents="lax"/>
  </xsd:complexType>

  <!-- The CCOpDataSpecType type is an enumerated type.
      The following strings are legal values for this type:
        LITERAL
        PARAMETER
        CACHE
        ASSOCIATION
        RANDOM
      An implementation may also recognize additional strings as extensions.
       -->
  <xsd:simpleType name="CCOpDataSpecType">
    <xsd:restriction base="xsd:string"/>
  </xsd:simpleType>

  <xsd:complexType name="CCOpReport">
    <xsd:sequence>
      <xsd:element name="data" type="xsd:string" minOccurs="0"/>
      <xsd:element name="opStatus" type="ale:CCStatus"/>
      <xsd:element name="opName" type="xsd:string" minOccurs="0"/>
      <xsd:element name="extension" type="ale:CCOpReportExtension" minOccurs="0"/>
      <xsd:any processContents="lax" minOccurs="0" maxOccurs="unbounded"
               namespace="##other"/>
    </xsd:sequence>
    <xsd:anyAttribute processContents="lax"/>
  </xsd:complexType>

  <xsd:complexType name="CCOpReportExtension">
    <xsd:sequence>
      <xsd:any processContents="lax" minOccurs="1" maxOccurs="unbounded"
               namespace="##local"/>
    </xsd:sequence>
    <xsd:anyAttribute processContents="lax"/>
  </xsd:complexType>

  <xsd:complexType name="CCOpSpec">
    <xsd:sequence>
      <xsd:element name="opType" type="ale:CCOpType"/>
      <xsd:element name="fieldspec" type="ale:ECFieldSpec" minOccurs="0"/>
      <xsd:element name="dataSpec" type="ale:CCOpDataSpec" minOccurs="0"/>
      <xsd:element name="opName" type="xsd:string" minOccurs="0"/>
      <xsd:element name="extension" type="ale:CCOpSpecExtension" minOccurs="0"/>
      <xsd:any processContents="lax" minOccurs="0" maxOccurs="unbounded"
               namespace="##other"/>
    </xsd:sequence>
    <xsd:anyAttribute processContents="lax"/>
  </xsd:complexType>

  <xsd:complexType name="CCOpSpecExtension">
    <xsd:sequence>
      <xsd:any processContents="lax" minOccurs="1" maxOccurs="unbounded"
               namespace="##local"/>
    </xsd:sequence>
    <xsd:anyAttribute processContents="lax"/>
  </xsd:complexType>

  <!-- The CCOpType type is an enumerated type.
       The following strings are legal values for this type:
         READ
         CHECK
         INITIALIZE
         ADD
         WRITE
         DELETE
         PASSWORD
         KILL
         LOCK
       An implementation may also recognize additional strings as extensions.
        -->
  <xsd:simpleType name="CCOpType">
    <xsd:restriction base="xsd:string"/>
  </xsd:simpleType>

  <xsd:complexType name="CCParameterList">
    <xsd:sequence>
      <xsd:element name="entries" minOccurs="0">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element name="entry" type="ale:CCParameterListEntry" minOccurs="0"
                         maxOccurs="unbounded"/>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
    </xsd:sequence>
  </xsd:complexType>

  <xsd:complexType name="CCParameterListEntry">
    <xsd:sequence>
      <xsd:element name="name" type="xsd:string"/>
      <xsd:element name="value" type="xsd:string"/>
    </xsd:sequence>
  </xsd:complexType>

  <xsd:complexType name="CCReports">
    <xsd:complexContent>
      <xsd:extension base="epcglobal:Document">
        <xsd:sequence>
          <xsd:element name="CCSpec" type="ale:CCSpec" minOccurs="0"/>
          <xsd:element name="cmdReports" minOccurs="0">
            <xsd:complexType>
              <xsd:sequence>
                <xsd:element name="cmdReport" type="ale:CCCmdReport" minOccurs="0"
                             maxOccurs="unbounded"/>
              </xsd:sequence>
            </xsd:complexType>
          </xsd:element>
          <xsd:element name="extension" type="ale:CCReportsExtension" minOccurs="0"/>
          <xsd:any processContents="lax" minOccurs="0" maxOccurs="unbounded"
                   namespace="##other"/>
        </xsd:sequence>
        <xsd:attribute name="specName" type="xsd:string" use="required"/>
        <xsd:attribute name="date" type="xsd:dateTime" use="required"/>
        <xsd:attribute name="ALEID" type="xsd:string" use="required"/>
        <xsd:attribute name="totalMilliseconds" type="xsd:long" use="required"/>
        <xsd:attribute name="initiationCondition" type="ale:CCInitiationCondition"
                       use="required"/>
        <xsd:attribute name="initiationTrigger" type="ale:ECTrigger" use="optional"/>
        <xsd:attribute name="terminationCondition" type="ale:CCTerminationCondition"
                       use="required"/>
        <xsd:attribute name="terminationTrigger" type="ale:ECTrigger" use="optional"/>
        <xsd:anyAttribute processContents="lax"/>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>

  <xsd:complexType name="CCReportsExtension">
    <xsd:sequence>
      <xsd:any processContents="lax" minOccurs="1" maxOccurs="unbounded"
               namespace="##local"/>
    </xsd:sequence>
    <xsd:anyAttribute processContents="lax"/>
  </xsd:complexType>

  <xsd:complexType name="CCSpec">
    <xsd:complexContent>
      <xsd:extension base="epcglobal:Document">
        <xsd:sequence>
          <xsd:element name="logicalReaders">
            <xsd:complexType>
              <xsd:sequence>
                <xsd:element name="logicalReader" type="xsd:string"
                             maxOccurs="unbounded"/>
              </xsd:sequence>
            </xsd:complexType>
          </xsd:element>
          <xsd:element name="boundarySpec" type="ale:CCBoundarySpec"/>
          <xsd:element name="cmdSpecs">
            <xsd:complexType>
              <xsd:sequence>
                <xsd:element name="cmdSpec" type="ale:CCCmdSpec" maxOccurs="unbounded"/>
              </xsd:sequence>
            </xsd:complexType>
          </xsd:element>
          <xsd:element name="extension" type="ale:CCSpecExtension" minOccurs="0"/>
          <xsd:any processContents="lax" minOccurs="0" maxOccurs="unbounded"
                   namespace="##other"/>
        </xsd:sequence>
        <xsd:attribute name="includeSpecInReports" type="xsd:boolean" default="false"/>
        <xsd:anyAttribute processContents="lax"/>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>

  <xsd:complexType name="CCSpecExtension">
    <xsd:sequence>
      <xsd:any processContents="lax" minOccurs="1" maxOccurs="unbounded"
               namespace="##local"/>
    </xsd:sequence>
    <xsd:anyAttribute processContents="lax"/>
  </xsd:complexType>

  <!-- The CCStatProfileName type is an enumerated type.
       The following strings are legal values for this type:
       An implementation may also recognize additional strings as extensions.
        -->
  <xsd:simpleType name="CCStatProfileName">
    <xsd:restriction base="xsd:string"/>
  </xsd:simpleType>

  <!-- The CCStatus type is an enumerated type.
       The following strings are legal values for this type:
         SUCCESS
         MISC_ERROR_TOTAL
         MISC_ERROR_PARTIAL
```

```
          PERMISSION_ERROR
          PASSWORD_ERROR
          FIELD_NOT_FOUND_ERROR
          OP_NOT_POSSIBLE_ERROR
          OUT_OF_RANGE_ERROR
          FIELD_EXISTS_ERROR
          MEMORY_OVERFLOW_ERROR
          MEMORY_CHECK_ERROR
          ASSOCIATION_TABLE_VALUE_INVALID
          ASSOCIATION_TABLE_VALUE_MISSING
          EPC_CACHE_DEPLETED
        An implementation may also recognize additional strings as extensions.
         -->
  <xsd:simpleType name="CCStatus">
    <xsd:restriction base="xsd:string"/>
  </xsd:simpleType>

  <xsd:complexType name="CCTagReport">
    <xsd:sequence>
      <xsd:element name="id" type="xsd:string" minOccurs="0"/>
      <xsd:element name="opReports" minOccurs="0">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element name="opReport" type="ale:CCOpReport" minOccurs="0"
                         maxOccurs="unbounded"/>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
      <xsd:element name="stats" minOccurs="0">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element name="stat" type="ale:CCTagStat" minOccurs="0"
                         maxOccurs="unbounded"/>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
      <xsd:element name="extension" type="ale:CCTagReportExtension" minOccurs="0"/>
      <xsd:any processContents="lax" minOccurs="0" maxOccurs="unbounded"
               namespace="##other"/>
    </xsd:sequence>
    <xsd:anyAttribute processContents="lax"/>
  </xsd:complexType>

  <xsd:complexType name="CCTagReportExtension">
    <xsd:sequence>
      <xsd:any processContents="lax" minOccurs="1" maxOccurs="unbounded"
               namespace="##local"/>
    </xsd:sequence>
    <xsd:anyAttribute processContents="lax"/>
  </xsd:complexType>

  <xsd:complexType name="CCTagStat">
    <xsd:sequence>
      <xsd:element name="profile" type="ale:CCStatProfileName"/>
      <xsd:element name="statBlocks" minOccurs="0">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element name="statBlock" type="ale:ECReaderStat" minOccurs="0"
                         maxOccurs="unbounded"/>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
    </xsd:sequence>
  </xsd:complexType>

  <!-- The CCTerminationCondition type is an enumerated type.
       The following strings are legal values for this type:
          TRIGGER
          DURATION
          NO_NEW_TAGS
          COUNT
          ERROR
          UNREQUEST
          UNDEFINE
        An implementation may also recognize additional strings as extensions.
         -->
  <xsd:simpleType name="CCTerminationCondition">
    <xsd:restriction base="xsd:string"/>
  </xsd:simpleType>

  <xsd:complexType name="EPCCacheSpec">
    <xsd:complexContent>
      <xsd:extension base="epcglobal:Document">
        <xsd:sequence>
          <xsd:element name="extension" type="ale:EPCCacheSpecExtension" minOccurs="0"/>
          <xsd:any processContents="lax" minOccurs="0" maxOccurs="unbounded"
                   namespace="##other"/>
        </xsd:sequence>
        <xsd:anyAttribute processContents="lax"/>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>

  <xsd:complexType name="EPCCacheSpecExtension">
    <xsd:sequence>
      <xsd:any processContents="lax" minOccurs="1" maxOccurs="unbounded"
               namespace="##local"/>
    </xsd:sequence>
    <xsd:anyAttribute processContents="lax"/>
  </xsd:complexType>

  <xsd:complexType name="EPCPatternList">
    <xsd:sequence>
      <xsd:element name="patterns" minOccurs="0">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element name="pattern" type="xsd:string" minOccurs="0"
                         maxOccurs="unbounded"/>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
    </xsd:sequence>
  </xsd:complexType>

  <xsd:complexType name="RNGSpec">
    <xsd:complexContent>
      <xsd:extension base="epcglobal:Document">
        <xsd:sequence>
          <xsd:element name="length" type="xsd:int"/>
          <xsd:element name="extension" type="ale:RNGSpecExtension" minOccurs="0"/>
          <xsd:any processContents="lax" minOccurs="0" maxOccurs="unbounded"
                   namespace="##other"/>
        </xsd:sequence>
        <xsd:anyAttribute processContents="lax"/>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>

  <xsd:complexType name="RNGSpecExtension">
    <xsd:sequence>
      <xsd:any processContents="lax" minOccurs="1" maxOccurs="unbounded"
               namespace="##local"/>
    </xsd:sequence>
    <xsd:anyAttribute processContents="lax"/>
  </xsd:complexType>
</xsd:schema>
```

3.7 ALE Tag Memory API Schema

The following is an XML Schema (XSD) for the ALE Tag Memory API, defining `TMSpec` as a top level element.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            xmlns:ale="urn:epcglobal:ale:xsd:1"
            targetNamespace="urn:epcglobal:ale:xsd:1"
            xmlns:epcglobal="urn:epcglobal:xsd:1" elementFormDefault="unqualified"
            attributeFormDefault="unqualified" version="1.0">
  <xsd:import namespace="urn:epcglobal:xsd:1" schemaLocation="EPCglobal.xsd"/>
  <!-- ALETM ELEMENTS -->
  <xsd:element name="TMSpec" type="ale:TMSpec"/>

  <!-- ALETM TYPES -->

  <xsd:complexType name="TMFixedFieldListSpec">
    <xsd:complexContent>
      <xsd:extension base="ale:TMSpec">
        <xsd:sequence>
          <xsd:element name="fixedFields" minOccurs="0">
            <xsd:complexType>
              <xsd:sequence>
                <xsd:element name="fixedField" type="ale:TMFixedFieldSpec"
                             minOccurs="0" maxOccurs="unbounded"/>
              </xsd:sequence>
            </xsd:complexType>
          </xsd:element>
          <xsd:element name="extension" type="ale:TMFixedFieldListSpecExtension"
                       minOccurs="0"/>
          <xsd:any processContents="lax" minOccurs="0" maxOccurs="unbounded"
                   namespace="##other"/>
        </xsd:sequence>
        <xsd:anyAttribute processContents="lax"/>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>

  <xsd:complexType name="TMFixedFieldListSpecExtension">
    <xsd:sequence>
      <xsd:any processContents="lax" minOccurs="1" maxOccurs="unbounded"
               namespace="##local"/>
    </xsd:sequence>
    <xsd:anyAttribute processContents="lax"/>
  </xsd:complexType>

  <xsd:complexType name="TMFixedFieldSpec">
    <xsd:sequence>
      <xsd:element name="fieldname" type="xsd:string"/>
      <xsd:element name="bank" type="xsd:int"/>
      <xsd:element name="length" type="xsd:int"/>
      <xsd:element name="offset" type="xsd:int"/>
      <xsd:element name="defaultDatatype" type="xsd:string"/>
      <xsd:element name="defaultFormat" type="xsd:string"/>
      <xsd:element name="extension" type="ale:TMFixedFieldSpecExtension" minOccurs="0"/>
      <xsd:any processContents="lax" minOccurs="0" maxOccurs="unbounded"
               namespace="##other"/>
    </xsd:sequence>
    <xsd:anyAttribute processContents="lax"/>
  </xsd:complexType>

  <xsd:complexType name="TMFixedFieldSpecExtension">
    <xsd:sequence>
      <xsd:any processContents="lax" minOccurs="1" maxOccurs="unbounded"
               namespace="##local"/>
    </xsd:sequence>
    <xsd:anyAttribute processContents="lax"/>
  </xsd:complexType>

  <xsd:complexType name="TMSpec" abstract="true">
    <xsd:complexContent>
      <xsd:extension base="epcglobal:Document">
        <xsd:sequence>
          <xsd:element name="baseExtension" type="ale:TMSpecExtension" minOccurs="0"/>
        </xsd:sequence>
        <xsd:anyAttribute processContents="lax"/>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>

  <xsd:complexType name="TMSpecExtension">
    <xsd:sequence>
      <xsd:any processContents="lax" minOccurs="1" maxOccurs="unbounded"
               namespace="##local"/>
    </xsd:sequence>
    <xsd:anyAttribute processContents="lax"/>
  </xsd:complexType>

  <xsd:complexType name="TMVariableFieldListSpec">
    <xsd:complexContent>
      <xsd:extension base="ale:TMSpec">
        <xsd:sequence>
          <xsd:element name="variableFields" minOccurs="0">
            <xsd:complexType>
              <xsd:sequence>
                <xsd:element name="variableField" type="ale:TMVariableFieldSpec"
                             minOccurs="0" maxOccurs="unbounded"/>
              </xsd:sequence>
            </xsd:complexType>
          </xsd:element>
          <xsd:element name="extension" type="ale:TMVariableFieldListSpecExtension"
                       minOccurs="0"/>
          <xsd:any processContents="lax" minOccurs="0" maxOccurs="unbounded"
                   namespace="##other"/>
        </xsd:sequence>
        <xsd:anyAttribute processContents="lax"/>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>

  <xsd:complexType name="TMVariableFieldListSpecExtension">
    <xsd:sequence>
      <xsd:any processContents="lax" minOccurs="1" maxOccurs="unbounded"
               namespace="##local"/>
    </xsd:sequence>
    <xsd:anyAttribute processContents="lax"/>
  </xsd:complexType>

  <xsd:complexType name="TMVariableFieldSpec">
    <xsd:sequence>
      <xsd:element name="fieldname" type="xsd:string"/>
      <xsd:element name="bank" type="xsd:int"/>
      <xsd:element name="oid" type="xsd:string"/>
      <xsd:element name="extension" type="ale:TMVariableFieldSpecExtension"
                   minOccurs="0"/>
      <xsd:any processContents="lax" minOccurs="0" maxOccurs="unbounded"
               namespace="##other"/>
    </xsd:sequence>
    <xsd:anyAttribute processContents="lax"/>
  </xsd:complexType>

  <xsd:complexType name="TMVariableFieldSpecExtension">
    <xsd:sequence>
      <xsd:any processContents="lax" minOccurs="1" maxOccurs="unbounded"
               namespace="##local"/>
    </xsd:sequence>
    <xsd:anyAttribute processContents="lax"/>
  </xsd:complexType>
```

```
</xsd:schema>
```

3.8 ALE Logical Reader API Schema

The following is an XML Schema (XSD) for the ALE Logical Reader API, defining `LRSpec`, and `LRProperty` as top level elements.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            xmlns:ale="urn:epcglobal:ale:xsd:1"
            targetNamespace="urn:epcglobal:ale:xsd:1"
            xmlns:epcglobal="urn:epcglobal:xsd:1" elementFormDefault="unqualified"
            attributeFormDefault="unqualified" version="1.0">
  <xsd:import namespace="urn:epcglobal:xsd:1" schemaLocation="EPCglobal.xsd"/>
  <!-- ALELR ELEMENTS -->
  <xsd:element name="LRSpec" type="ale:LRSpec"/>
  <xsd:element name="LRProperty" type="ale:LRProperty"/>

  <!-- ALELR TYPES -->

  <xsd:complexType name="LRProperty">
    <xsd:sequence>
      <xsd:element name="name" type="xsd:string"/>
      <xsd:element name="value" type="xsd:string" minOccurs="0"/>
    </xsd:sequence>
  </xsd:complexType>

  <xsd:complexType name="LRSpec">
    <xsd:complexContent>
      <xsd:extension base="epcglobal:Document">
        <xsd:sequence>
          <xsd:element name="isComposite" type="xsd:boolean" minOccurs="0"/>
          <xsd:element name="readers" minOccurs="0">
            <xsd:complexType>
              <xsd:sequence>
                <xsd:element name="reader" type="xsd:string" minOccurs="0"
                             maxOccurs="unbounded"/>
              </xsd:sequence>
            </xsd:complexType>
          </xsd:element>
          <xsd:element name="properties" minOccurs="0">
            <xsd:complexType>
              <xsd:sequence>
                <xsd:element name="property" type="ale:LRProperty" minOccurs="0"
                             maxOccurs="unbounded"/>
              </xsd:sequence>
            </xsd:complexType>
          </xsd:element>
          <xsd:element name="extension" type="ale:LRSpecExtension" minOccurs="0"/>
          <xsd:any processContents="lax" minOccurs="0" maxOccurs="unbounded"
                   namespace="##other"/>
        </xsd:sequence>
        <xsd:anyAttribute processContents="lax"/>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>

  <xsd:complexType name="LRSpecExtension">
    <xsd:sequence>
      <xsd:any processContents="lax" minOccurs="1" maxOccurs="unbounded"
               namespace="##local"/>
    </xsd:sequence>
    <xsd:anyAttribute processContents="lax"/>
  </xsd:complexType>
</xsd:schema>
```

3.9 ALE Access Control API Schema

The following is an XML Schema (XSD) for the ALE Access Control API, defining `ACPermission`, `ACRole`, and `ACClientIdentity` as top level elements.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            xmlns:ale="urn:epcglobal:ale:xsd:1"
            targetNamespace="urn:epcglobal:ale:xsd:1"
            xmlns:epcglobal="urn:epcglobal:xsd:1" elementFormDefault="unqualified"
            attributeFormDefault="unqualified" version="1.0">
  <xsd:import namespace="urn:epcglobal:xsd:1" schemaLocation="EPCglobal.xsd"/>
  <!-- ALEAC ELEMENTS -->
  <xsd:element name="ACPermission" type="ale:ACPermission"/>
  <xsd:element name="ACRole" type="ale:ACRole"/>
  <xsd:element name="ACClientIdentity" type="ale:ACClientIdentity"/>

  <!-- ALEAC TYPES -->

  <!-- The ACClass type is an enumerated type.
       The following strings are legal values for this type:
         Method
       An implementation may also recognize additional strings as extensions.
        -->
  <xsd:simpleType name="ACClass">
    <xsd:restriction base="xsd:string"/>
  </xsd:simpleType>

  <xsd:complexType name="ACClientCredential">
    <xsd:sequence>
      <xsd:element name="extension" type="ale:ACClientCredentialExtension"
                   minOccurs="0"/>
      <xsd:any processContents="lax" minOccurs="0" maxOccurs="unbounded"
               namespace="##other"/>
    </xsd:sequence>
    <xsd:anyAttribute processContents="lax"/>
  </xsd:complexType>

  <xsd:complexType name="ACClientCredentialExtension">
    <xsd:sequence>
      <xsd:any processContents="lax" minOccurs="1" maxOccurs="unbounded"
               namespace="##local"/>
    </xsd:sequence>
    <xsd:anyAttribute processContents="lax"/>
  </xsd:complexType>

  <xsd:complexType name="ACClientIdentity">
    <xsd:complexContent>
      <xsd:extension base="epcglobal:Document">
        <xsd:sequence>
          <xsd:element name="credentials" minOccurs="0">
            <xsd:complexType>
              <xsd:sequence>
                <xsd:element name="credential" type="ale:ACClientCredential"
                             minOccurs="0" maxOccurs="unbounded"/>
              </xsd:sequence>
            </xsd:complexType>
          </xsd:element>
          <xsd:element name="roleNames" minOccurs="0">
            <xsd:complexType>
              <xsd:sequence>
                <xsd:element name="roleName" type="xsd:string" minOccurs="0"
                             maxOccurs="unbounded"/>
              </xsd:sequence>
            </xsd:complexType>
          </xsd:element>
          <xsd:element name="extension" type="ale:ACClientIdentityExtension"
                       minOccurs="0"/>
          <xsd:any processContents="lax" minOccurs="0" maxOccurs="unbounded"
```

```
                 namespace="##other"/>
      </xsd:sequence>
      <xsd:anyAttribute processContents="lax"/>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="ACClientIdentityExtension">
  <xsd:sequence>
    <xsd:any processContents="lax" minOccurs="1" maxOccurs="unbounded"
             namespace="##local"/>
  </xsd:sequence>
  <xsd:anyAttribute processContents="lax"/>
</xsd:complexType>

<xsd:complexType name="ACPermission">
  <xsd:complexContent>
    <xsd:extension base="epcglobal:Document">
      <xsd:sequence>
        <xsd:element name="permissionClass" type="ale:ACClass"/>
        <xsd:element name="instances" minOccurs="0">
          <xsd:complexType>
            <xsd:sequence>
              <xsd:element name="instance" type="xsd:string" minOccurs="0"
                           maxOccurs="unbounded"/>
            </xsd:sequence>
          </xsd:complexType>
        </xsd:element>
        <xsd:element name="extension" type="ale:ACPermissionExtension" minOccurs="0"/>
        <xsd:any processContents="lax" minOccurs="0" maxOccurs="unbounded"
                 namespace="##other"/>
      </xsd:sequence>
      <xsd:anyAttribute processContents="lax"/>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="ACPermissionExtension">
  <xsd:sequence>
    <xsd:any processContents="lax" minOccurs="1" maxOccurs="unbounded"
             namespace="##local"/>
  </xsd:sequence>
  <xsd:anyAttribute processContents="lax"/>
</xsd:complexType>

<xsd:complexType name="ACRole">
  <xsd:complexContent>
    <xsd:extension base="epcglobal:Document">
      <xsd:sequence>
        <xsd:element name="permissionNames" minOccurs="0">
          <xsd:complexType>
            <xsd:sequence>
              <xsd:element name="permissionName" type="xsd:string" minOccurs="0"
                           maxOccurs="unbounded"/>
            </xsd:sequence>
          </xsd:complexType>
        </xsd:element>
        <xsd:element name="extension" type="ale:ACRoleExtension" minOccurs="0"/>
        <xsd:any processContents="lax" minOccurs="0" maxOccurs="unbounded"
                 namespace="##other"/>
      </xsd:sequence>
      <xsd:anyAttribute processContents="lax"/>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="ACRoleExtension">
  <xsd:sequence>
    <xsd:any processContents="lax" minOccurs="1" maxOccurs="unbounded"
             namespace="##local"/>
    </xsd:sequence>
    <xsd:anyAttribute processContents="lax"/>
  </xsd:complexType>
</xsd:schema>
```

4 SOAP Bindings

This section defines SOAP 1.1 [SOAP1.1] bindings for the five ALE APIs, using WSDL. All SOAP bindings are WS-I Basic Profile 1.0 [WSI] compliant for greatest interoperability.

4.1 Organization of the SOAP Bindings

Because an implementation may not implement all five ALE APIs, each API has a separate WSDL file. The request, response, and exception message types are defined in a separate XML namespace for each API. Complex data types referred to in requests and responses are all imported from the XML schemas defined in Section 2, which are all in a common XML namespace..

The five WSDL files are summarized in the table below:

WSDL	Section	Imported Schema	XML Namespace
`ale`	4.3	`ale`	`urn:epcglobal:ale:wsdl:1`
`alecc`	4.4	`alecc`	`urn:epcglobal:alecc:wsdl:1`
`aletm`	4.5	`aletm`	`urn:epcglobal:aletm:wsdl:1`
`alelr`	4.6	`alelr`	`urn:epcglobal:alelr:wsdl:1`
`aleac`	4.7	`aleac`	`urn:epcglobal:aleac:wsdl:1`

4.2 Link Level Security

ALE implementations that wish to provide a measure of security between themselves and their clients MAY implement an HTTPS ("HTTP Over TLS") [RFC2818] transport for SOAP messages.

4.3 ALE Reading API SOAP Binding

The following is a Web Services Description Language (WSDL) 1.1 [WSDL1.1] specification defining the standard SOAP 1.1 [SOAP1.1] binding of the ALE Reading API. This SOAP binding is compliant with the WS-I Basic Profile Version 1.0 [WSI].

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- ALESERVICE DEFINITIONS -->
<wsdl:definitions xmlns:xsd="http://www.w3.org/2001/XMLSchema"
                  xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
                  xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/"
                  xmlns:ale="urn:epcglobal:ale:xsd:1"
                  xmlns:impl="urn:epcglobal:ale:wsdl:1"
                  targetNamespace="urn:epcglobal:ale:wsdl:1">
  <!-- ALESERVICE TYPES -->
  <wsdl:types>
```

```
<xsd:schema targetNamespace="urn:epcglobal:ale:wsdl:1">
  <xsd:import namespace="urn:epcglobal:ale:xsd:1"
              schemaLocation="EPCglobal-ale-1_1-ale.xsd"/>
  <!-- ALESERVICE MESSAGE WRAPPERS -->

  <xsd:element name="Define" type="impl:Define"/>
  <xsd:complexType name="Define">
    <xsd:sequence>
      <xsd:element name="specName" type="xsd:string"/>
      <xsd:element name="spec" type="ale:ECSpec"/>
    </xsd:sequence>
  </xsd:complexType>

  <xsd:element name="Undefine" type="impl:Undefine"/>
  <xsd:complexType name="Undefine">
    <xsd:sequence>
      <xsd:element name="specName" type="xsd:string"/>
    </xsd:sequence>
  </xsd:complexType>

  <xsd:element name="GetECSpec" type="impl:GetECSpec"/>
  <xsd:complexType name="GetECSpec">
    <xsd:sequence>
      <xsd:element name="specName" type="xsd:string"/>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:element name="GetECSpecResult" type="ale:ECSpec"/>

  <xsd:element name="GetECSpecNames" type="impl:EmptyParms"/>
  <xsd:element name="GetECSpecNamesResult" type="impl:ArrayOfString"/>

  <xsd:element name="Subscribe" type="impl:Subscribe"/>
  <xsd:complexType name="Subscribe">
    <xsd:sequence>
      <xsd:element name="specName" type="xsd:string"/>
      <xsd:element name="notificationURI" type="xsd:string"/>
    </xsd:sequence>
  </xsd:complexType>

  <xsd:element name="Unsubscribe" type="impl:Unsubscribe"/>
  <xsd:complexType name="Unsubscribe">
    <xsd:sequence>
      <xsd:element name="specName" type="xsd:string"/>
      <xsd:element name="notificationURI" type="xsd:string"/>
    </xsd:sequence>
  </xsd:complexType>

  <xsd:element name="Poll" type="impl:Poll"/>
  <xsd:complexType name="Poll">
    <xsd:sequence>
      <xsd:element name="specName" type="xsd:string"/>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:element name="PollResult" type="ale:ECReports"/>

  <xsd:element name="Immediate" type="impl:Immediate"/>
  <xsd:complexType name="Immediate">
    <xsd:sequence>
      <xsd:element name="spec" type="ale:ECSpec"/>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:element name="ImmediateResult" type="ale:ECReports"/>

  <xsd:element name="GetSubscribers" type="impl:GetSubscribers"/>
  <xsd:complexType name="GetSubscribers">
    <xsd:sequence>
      <xsd:element name="specName" type="xsd:string"/>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:element name="GetSubscribersResult" type="impl:ArrayOfString"/>
```

```

<xsd:element name="GetStandardVersion" type="impl:EmptyParms"/>
<xsd:element name="GetStandardVersionResult" type="xsd:string"/>

<xsd:element name="GetVendorVersion" type="impl:EmptyParms"/>
<xsd:element name="GetVendorVersionResult" type="xsd:string"/>

<xsd:element name="ALEException" type="impl:ALEException"/>
<xsd:complexType name="ALEException">
  <xsd:sequence>
    <xsd:element name="reason" type="xsd:string"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:element name="SecurityException" type="impl:SecurityException"/>
<xsd:complexType name="SecurityException">
  <xsd:complexContent>
    <xsd:extension base="impl:ALEException"/>
  </xsd:complexContent>
</xsd:complexType>

<xsd:element name="DuplicateNameException" type="impl:DuplicateNameException"/>
<xsd:complexType name="DuplicateNameException">
  <xsd:complexContent>
    <xsd:extension base="impl:ALEException"/>
  </xsd:complexContent>
</xsd:complexType>

<xsd:element name="ECSpecValidationException"
             type="impl:ECSpecValidationException"/>
<xsd:complexType name="ECSpecValidationException">
  <xsd:complexContent>
    <xsd:extension base="impl:ALEException"/>
  </xsd:complexContent>
</xsd:complexType>

<xsd:element name="InvalidURIException" type="impl:InvalidURIException"/>
<xsd:complexType name="InvalidURIException">
  <xsd:complexContent>
    <xsd:extension base="impl:ALEException"/>
  </xsd:complexContent>
</xsd:complexType>

<xsd:element name="NoSuchNameException" type="impl:NoSuchNameException"/>
<xsd:complexType name="NoSuchNameException">
  <xsd:complexContent>
    <xsd:extension base="impl:ALEException"/>
  </xsd:complexContent>
</xsd:complexType>

<xsd:element name="NoSuchSubscriberException"
             type="impl:NoSuchSubscriberException"/>
<xsd:complexType name="NoSuchSubscriberException">
  <xsd:complexContent>
    <xsd:extension base="impl:ALEException"/>
  </xsd:complexContent>
</xsd:complexType>

<xsd:element name="DuplicateSubscriptionException"
             type="impl:DuplicateSubscriptionException"/>
<xsd:complexType name="DuplicateSubscriptionException">
  <xsd:complexContent>
    <xsd:extension base="impl:ALEException"/>
  </xsd:complexContent>
</xsd:complexType>

<xsd:element name="ImplementationException" type="impl:ImplementationException"/>
<xsd:complexType name="ImplementationException">
  <xsd:complexContent>
    <xsd:extension base="impl:ALEException">
```

```
          <xsd:sequence>
            <xsd:element name="severity" type="impl:ImplementationExceptionSeverity"/>
          </xsd:sequence>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="ArrayOfString">
      <xsd:sequence>
        <xsd:element name="string" type="xsd:string" minOccurs="0"
                     maxOccurs="unbounded"/>
      </xsd:sequence>
    </xsd:complexType>

    <!-- The ImplementationExceptionSeverity type is an enumerated type.
         The following strings are legal values for this type:
          ERROR
          SEVERE
         -->
    <xsd:simpleType name="ImplementationExceptionSeverity">
      <xsd:restriction base="xsd:string"/>
    </xsd:simpleType>

    <xsd:element name="VoidHolder" type="impl:VoidHolder"/>
    <xsd:complexType name="VoidHolder"/>

    <xsd:complexType name="EmptyParms"/>
  </xsd:schema>
</wsdl:types>
<!-- ALESERVICE MESSAGES -->

<wsdl:message name="defineRequest">
  <wsdl:part name="parms" element="impl:Define"/>
</wsdl:message>
<wsdl:message name="defineResponse">
  <wsdl:part name="defineReturn" element="impl:VoidHolder"/>
</wsdl:message>

<wsdl:message name="undefineRequest">
  <wsdl:part name="parms" element="impl:Undefine"/>
</wsdl:message>
<wsdl:message name="undefineResponse">
  <wsdl:part name="undefineReturn" element="impl:VoidHolder"/>
</wsdl:message>

<wsdl:message name="getECSpecRequest">
  <wsdl:part name="parms" element="impl:GetECSpec"/>
</wsdl:message>
<wsdl:message name="getECSpecResponse">
  <wsdl:part name="getECSpecReturn" element="impl:GetECSpecResult"/>
</wsdl:message>

<wsdl:message name="getECSpecNamesRequest">
  <wsdl:part name="parms" element="impl:GetECSpecNames"/>
</wsdl:message>
<wsdl:message name="getECSpecNamesResponse">
  <wsdl:part name="getECSpecNamesReturn" element="impl:GetECSpecNamesResult"/>
</wsdl:message>

<wsdl:message name="subscribeRequest">
  <wsdl:part name="parms" element="impl:Subscribe"/>
</wsdl:message>
<wsdl:message name="subscribeResponse">
  <wsdl:part name="subscribeReturn" element="impl:VoidHolder"/>
</wsdl:message>

<wsdl:message name="unsubscribeRequest">
  <wsdl:part name="parms" element="impl:Unsubscribe"/>
</wsdl:message>
<wsdl:message name="unsubscribeResponse">
```

```
    <wsdl:part name="unsubscribeReturn" element="impl:VoidHolder"/>
  </wsdl:message>

  <wsdl:message name="pollRequest">
    <wsdl:part name="parms" element="impl:Poll"/>
  </wsdl:message>
  <wsdl:message name="pollResponse">
    <wsdl:part name="pollReturn" element="impl:PollResult"/>
  </wsdl:message>

  <wsdl:message name="immediateRequest">
    <wsdl:part name="parms" element="impl:Immediate"/>
  </wsdl:message>
  <wsdl:message name="immediateResponse">
    <wsdl:part name="immediateReturn" element="impl:ImmediateResult"/>
  </wsdl:message>

  <wsdl:message name="getSubscribersRequest">
    <wsdl:part name="parms" element="impl:GetSubscribers"/>
  </wsdl:message>
  <wsdl:message name="getSubscribersResponse">
    <wsdl:part name="getSubscribersReturn" element="impl:GetSubscribersResult"/>
  </wsdl:message>

  <wsdl:message name="getStandardVersionRequest">
    <wsdl:part name="parms" element="impl:GetStandardVersion"/>
  </wsdl:message>
  <wsdl:message name="getStandardVersionResponse">
    <wsdl:part name="getStandardVersionReturn" element="impl:GetStandardVersionResult"/>
  </wsdl:message>

  <wsdl:message name="getVendorVersionRequest">
    <wsdl:part name="parms" element="impl:GetVendorVersion"/>
  </wsdl:message>
  <wsdl:message name="getVendorVersionResponse">
    <wsdl:part name="getVendorVersionReturn" element="impl:GetVendorVersionResult"/>
  </wsdl:message>

  <wsdl:message name="SecurityExceptionResponse">
    <wsdl:part name="fault" element="impl:SecurityException"/>
  </wsdl:message>

  <wsdl:message name="DuplicateNameExceptionResponse">
    <wsdl:part name="fault" element="impl:DuplicateNameException"/>
  </wsdl:message>

  <wsdl:message name="ECSpecValidationExceptionResponse">
    <wsdl:part name="fault" element="impl:ECSpecValidationException"/>
  </wsdl:message>

  <wsdl:message name="InvalidURIExceptionResponse">
    <wsdl:part name="fault" element="impl:InvalidURIException"/>
  </wsdl:message>

  <wsdl:message name="NoSuchNameExceptionResponse">
    <wsdl:part name="fault" element="impl:NoSuchNameException"/>
  </wsdl:message>

  <wsdl:message name="NoSuchSubscriberExceptionResponse">
    <wsdl:part name="fault" element="impl:NoSuchSubscriberException"/>
  </wsdl:message>

  <wsdl:message name="DuplicateSubscriptionExceptionResponse">
    <wsdl:part name="fault" element="impl:DuplicateSubscriptionException"/>
  </wsdl:message>

  <wsdl:message name="ImplementationExceptionResponse">
    <wsdl:part name="fault" element="impl:ImplementationException"/>
  </wsdl:message>
  <!-- ALESERVICE PORTTYPE -->
```

```
    <wsdl:portType name="ALEServicePortType">

      <wsdl:operation name="define">
        <wsdl:input message="impl:defineRequest" name="defineRequest"/>
        <wsdl:output message="impl:defineResponse" name="defineResponse"/>
        <wsdl:fault message="impl:DuplicateNameExceptionResponse"
                    name="DuplicateNameExceptionFault"/>
        <wsdl:fault message="impl:ECSpecValidationExceptionResponse"
                    name="ECSpecValidationExceptionFault"/>
        <wsdl:fault message="impl:SecurityExceptionResponse"
                    name="SecurityExceptionFault"/>
        <wsdl:fault message="impl:ImplementationExceptionResponse"
                    name="ImplementationExceptionFault"/>
      </wsdl:operation>

      <wsdl:operation name="undefine">
        <wsdl:input message="impl:undefineRequest" name="undefineRequest"/>
        <wsdl:output message="impl:undefineResponse" name="undefineResponse"/>
        <wsdl:fault message="impl:NoSuchNameExceptionResponse"
                    name="NoSuchNameExceptionFault"/>
        <wsdl:fault message="impl:SecurityExceptionResponse"
                    name="SecurityExceptionFault"/>
        <wsdl:fault message="impl:ImplementationExceptionResponse"
                    name="ImplementationExceptionFault"/>
      </wsdl:operation>

      <wsdl:operation name="getECSpec">
        <wsdl:input message="impl:getECSpecRequest" name="getECSpecRequest"/>
        <wsdl:output message="impl:getECSpecResponse" name="getECSpecResponse"/>
        <wsdl:fault message="impl:NoSuchNameExceptionResponse"
                    name="NoSuchNameExceptionFault"/>
        <wsdl:fault message="impl:SecurityExceptionResponse"
                    name="SecurityExceptionFault"/>
        <wsdl:fault message="impl:ImplementationExceptionResponse"
                    name="ImplementationExceptionFault"/>
      </wsdl:operation>

      <wsdl:operation name="getECSpecNames">
        <wsdl:input message="impl:getECSpecNamesRequest" name="getECSpecNamesRequest"/>
        <wsdl:output message="impl:getECSpecNamesResponse" name="getECSpecNamesResponse"/>
        <wsdl:fault message="impl:SecurityExceptionResponse"
                    name="SecurityExceptionFault"/>
        <wsdl:fault message="impl:ImplementationExceptionResponse"
                    name="ImplementationExceptionFault"/>
      </wsdl:operation>

      <wsdl:operation name="subscribe">
        <wsdl:input message="impl:subscribeRequest" name="subscribeRequest"/>
        <wsdl:output message="impl:subscribeResponse" name="subscribeResponse"/>
        <wsdl:fault message="impl:NoSuchNameExceptionResponse"
                    name="NoSuchNameExceptionFault"/>
        <wsdl:fault message="impl:InvalidURIExceptionResponse"
                    name="InvalidURIExceptionFault"/>
        <wsdl:fault message="impl:DuplicateSubscriptionExceptionResponse"
                    name="DuplicateSubscriptionExceptionFault"/>
        <wsdl:fault message="impl:SecurityExceptionResponse"
                    name="SecurityExceptionFault"/>
        <wsdl:fault message="impl:ImplementationExceptionResponse"
                    name="ImplementationExceptionFault"/>
      </wsdl:operation>

      <wsdl:operation name="unsubscribe">
        <wsdl:input message="impl:unsubscribeRequest" name="unsubscribeRequest"/>
        <wsdl:output message="impl:unsubscribeResponse" name="unsubscribeResponse"/>
        <wsdl:fault message="impl:NoSuchNameExceptionResponse"
                    name="NoSuchNameExceptionFault"/>
        <wsdl:fault message="impl:NoSuchSubscriberExceptionResponse"
                    name="NoSuchSubscriberExceptionFault"/>
        <wsdl:fault message="impl:InvalidURIExceptionResponse"
```

```
                  name="InvalidURIExceptionFault"/>
      <wsdl:fault message="impl:SecurityExceptionResponse"
                  name="SecurityExceptionFault"/>
      <wsdl:fault message="impl:ImplementationExceptionResponse"
                  name="ImplementationExceptionFault"/>
    </wsdl:operation>

    <wsdl:operation name="poll">
      <wsdl:input message="impl:pollRequest" name="pollRequest"/>
      <wsdl:output message="impl:pollResponse" name="pollResponse"/>
      <wsdl:fault message="impl:NoSuchNameExceptionResponse"
                  name="NoSuchNameExceptionFault"/>
      <wsdl:fault message="impl:SecurityExceptionResponse"
                  name="SecurityExceptionFault"/>
      <wsdl:fault message="impl:ImplementationExceptionResponse"
                  name="ImplementationExceptionFault"/>
    </wsdl:operation>

    <wsdl:operation name="immediate">
      <wsdl:input message="impl:immediateRequest" name="immediateRequest"/>
      <wsdl:output message="impl:immediateResponse" name="immediateResponse"/>
      <wsdl:fault message="impl:ECSpecValidationExceptionResponse"
                  name="ECSpecValidationExceptionFault"/>
      <wsdl:fault message="impl:SecurityExceptionResponse"
                  name="SecurityExceptionFault"/>
      <wsdl:fault message="impl:ImplementationExceptionResponse"
                  name="ImplementationExceptionFault"/>
    </wsdl:operation>

    <wsdl:operation name="getSubscribers">
      <wsdl:input message="impl:getSubscribersRequest" name="getSubscribersRequest"/>
      <wsdl:output message="impl:getSubscribersResponse" name="getSubscribersResponse"/>
      <wsdl:fault message="impl:NoSuchNameExceptionResponse"
                  name="NoSuchNameExceptionFault"/>
      <wsdl:fault message="impl:SecurityExceptionResponse"
                  name="SecurityExceptionFault"/>
      <wsdl:fault message="impl:ImplementationExceptionResponse"
                  name="ImplementationExceptionFault"/>
    </wsdl:operation>

    <wsdl:operation name="getStandardVersion">
      <wsdl:input message="impl:getStandardVersionRequest"
                  name="getStandardVersionRequest"/>
      <wsdl:output message="impl:getStandardVersionResponse"
                   name="getStandardVersionResponse"/>
      <wsdl:fault message="impl:ImplementationExceptionResponse"
                  name="ImplementationExceptionFault"/>
    </wsdl:operation>

    <wsdl:operation name="getVendorVersion">
      <wsdl:input message="impl:getVendorVersionRequest" name="getVendorVersionRequest"/>
      <wsdl:output message="impl:getVendorVersionResponse"
                   name="getVendorVersionResponse"/>
      <wsdl:fault message="impl:ImplementationExceptionResponse"
                  name="ImplementationExceptionFault"/>
    </wsdl:operation>
  </wsdl:portType>
  <!-- ALESERVICE BINDING -->

  <wsdl:binding name="ALEServiceBinding" type="impl:ALEServicePortType">
    <wsdlsoap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>

    <wsdl:operation name="define">
      <wsdlsoap:operation soapAction=""/>
      <wsdl:input name="defineRequest">
        <wsdlsoap:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="defineResponse">
        <wsdlsoap:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="DuplicateNameExceptionFault">
        <wsdlsoap:fault name="DuplicateNameExceptionFault" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="ECSpecValidationExceptionFault">
        <wsdlsoap:fault name="ECSpecValidationExceptionFault" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="SecurityExceptionFault">
        <wsdlsoap:fault name="SecurityExceptionFault" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="ImplementationExceptionFault">
        <wsdlsoap:fault name="ImplementationExceptionFault" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>

    <wsdl:operation name="undefine">
      <wsdlsoap:operation soapAction=""/>
      <wsdl:input name="undefineRequest">
        <wsdlsoap:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="undefineResponse">
        <wsdlsoap:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="NoSuchNameExceptionFault">
        <wsdlsoap:fault name="NoSuchNameExceptionFault" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="SecurityExceptionFault">
        <wsdlsoap:fault name="SecurityExceptionFault" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="ImplementationExceptionFault">
        <wsdlsoap:fault name="ImplementationExceptionFault" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>

    <wsdl:operation name="getECSpec">
      <wsdlsoap:operation soapAction=""/>
      <wsdl:input name="getECSpecRequest">
        <wsdlsoap:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="getECSpecResponse">
        <wsdlsoap:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="NoSuchNameExceptionFault">
        <wsdlsoap:fault name="NoSuchNameExceptionFault" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="SecurityExceptionFault">
        <wsdlsoap:fault name="SecurityExceptionFault" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="ImplementationExceptionFault">
        <wsdlsoap:fault name="ImplementationExceptionFault" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>

    <wsdl:operation name="getECSpecNames">
      <wsdlsoap:operation soapAction=""/>
      <wsdl:input name="getECSpecNamesRequest">
        <wsdlsoap:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="getECSpecNamesResponse">
        <wsdlsoap:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="SecurityExceptionFault">
        <wsdlsoap:fault name="SecurityExceptionFault" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="ImplementationExceptionFault">
        <wsdlsoap:fault name="ImplementationExceptionFault" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>

    <wsdl:operation name="subscribe">
      <wsdlsoap:operation soapAction=""/>
      <wsdl:input name="subscribeRequest">
        <wsdlsoap:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="subscribeResponse">
        <wsdlsoap:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="NoSuchNameExceptionFault">
        <wsdlsoap:fault name="NoSuchNameExceptionFault" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="InvalidURIExceptionFault">
        <wsdlsoap:fault name="InvalidURIExceptionFault" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="DuplicateSubscriptionExceptionFault">
        <wsdlsoap:fault name="DuplicateSubscriptionExceptionFault" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="SecurityExceptionFault">
        <wsdlsoap:fault name="SecurityExceptionFault" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="ImplementationExceptionFault">
        <wsdlsoap:fault name="ImplementationExceptionFault" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>

    <wsdl:operation name="unsubscribe">
      <wsdlsoap:operation soapAction=""/>
      <wsdl:input name="unsubscribeRequest">
        <wsdlsoap:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="unsubscribeResponse">
        <wsdlsoap:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="NoSuchNameExceptionFault">
        <wsdlsoap:fault name="NoSuchNameExceptionFault" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="NoSuchSubscriberExceptionFault">
        <wsdlsoap:fault name="NoSuchSubscriberExceptionFault" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="InvalidURIExceptionFault">
        <wsdlsoap:fault name="InvalidURIExceptionFault" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="SecurityExceptionFault">
        <wsdlsoap:fault name="SecurityExceptionFault" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="ImplementationExceptionFault">
        <wsdlsoap:fault name="ImplementationExceptionFault" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>

    <wsdl:operation name="poll">
      <wsdlsoap:operation soapAction=""/>
      <wsdl:input name="pollRequest">
        <wsdlsoap:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="pollResponse">
        <wsdlsoap:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="NoSuchNameExceptionFault">
        <wsdlsoap:fault name="NoSuchNameExceptionFault" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="SecurityExceptionFault">
        <wsdlsoap:fault name="SecurityExceptionFault" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="ImplementationExceptionFault">
        <wsdlsoap:fault name="ImplementationExceptionFault" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>

    <wsdl:operation name="immediate">
      <wsdlsoap:operation soapAction=""/>
      <wsdl:input name="immediateRequest">
```

```
          <wsdlsoap:body use="literal"/>
        </wsdl:input>
        <wsdl:output name="immediateResponse">
          <wsdlsoap:body use="literal"/>
        </wsdl:output>
        <wsdl:fault name="ECSpecValidationExceptionFault">
          <wsdlsoap:fault name="ECSpecValidationExceptionFault" use="literal"/>
        </wsdl:fault>
        <wsdl:fault name="SecurityExceptionFault">
          <wsdlsoap:fault name="SecurityExceptionFault" use="literal"/>
        </wsdl:fault>
        <wsdl:fault name="ImplementationExceptionFault">
          <wsdlsoap:fault name="ImplementationExceptionFault" use="literal"/>
        </wsdl:fault>
      </wsdl:operation>

      <wsdl:operation name="getSubscribers">
        <wsdlsoap:operation soapAction=""/>
        <wsdl:input name="getSubscribersRequest">
          <wsdlsoap:body use="literal"/>
        </wsdl:input>
        <wsdl:output name="getSubscribersResponse">
          <wsdlsoap:body use="literal"/>
        </wsdl:output>
        <wsdl:fault name="NoSuchNameExceptionFault">
          <wsdlsoap:fault name="NoSuchNameExceptionFault" use="literal"/>
        </wsdl:fault>
        <wsdl:fault name="SecurityExceptionFault">
          <wsdlsoap:fault name="SecurityExceptionFault" use="literal"/>
        </wsdl:fault>
        <wsdl:fault name="ImplementationExceptionFault">
          <wsdlsoap:fault name="ImplementationExceptionFault" use="literal"/>
        </wsdl:fault>
      </wsdl:operation>

      <wsdl:operation name="getStandardVersion">
        <wsdlsoap:operation soapAction=""/>
        <wsdl:input name="getStandardVersionRequest">
          <wsdlsoap:body use="literal"/>
        </wsdl:input>
        <wsdl:output name="getStandardVersionResponse">
          <wsdlsoap:body use="literal"/>
        </wsdl:output>
        <wsdl:fault name="ImplementationExceptionFault">
          <wsdlsoap:fault name="ImplementationExceptionFault" use="literal"/>
        </wsdl:fault>
      </wsdl:operation>

      <wsdl:operation name="getVendorVersion">
        <wsdlsoap:operation soapAction=""/>
        <wsdl:input name="getVendorVersionRequest">
          <wsdlsoap:body use="literal"/>
        </wsdl:input>
        <wsdl:output name="getVendorVersionResponse">
          <wsdlsoap:body use="literal"/>
        </wsdl:output>
        <wsdl:fault name="ImplementationExceptionFault">
          <wsdlsoap:fault name="ImplementationExceptionFault" use="literal"/>
        </wsdl:fault>
      </wsdl:operation>
    </wsdl:binding>

    <!-- ALESERVICE -->
    <wsdl:service name="ALEService">
      <wsdl:port binding="impl:ALEServiceBinding" name="ALEServicePort">
        <!-- The value of the location attribute below is an example only;
             Implementations are free to choose any appropriate URL. -->
        <wsdlsoap:address location="http://localhost:8080/services/ALEService"/>
      </wsdl:port>
    </wsdl:service>
```

```
</wsdl:definitions>
```

4.4 ALE Writing API SOAP Binding

The following is a Web Services Description Language (WSDL) 1.1 [WSDL1.1] specification defining the standard SOAP 1.1 [SOAP1.1] binding of the ALE Writing API. This SOAP binding is compliant with the WS-I Basic Profile Version 1.0 [WSI].

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- ALECCSERVICE DEFINITIONS -->
<wsdl:definitions xmlns:xsd="http://www.w3.org/2001/XMLSchema"
                  xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
                  xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/"
                  xmlns:ale="urn:epcglobal:ale:xsd:1"
                  xmlns:impl="urn:epcglobal:alecc:wsdl:1"
                  targetNamespace="urn:epcglobal:alecc:wsdl:1">
  <!-- ALECCSERVICE TYPES -->
  <wsdl:types>
    <xsd:schema targetNamespace="urn:epcglobal:alecc:wsdl:1">
      <xsd:import namespace="urn:epcglobal:ale:xsd:1"
                  schemaLocation="EPCglobal-ale-1_1-alecc.xsd"/>
      <!-- ALECCSERVICE MESSAGE WRAPPERS -->

      <xsd:element name="Define" type="impl:Define"/>
      <xsd:complexType name="Define">
        <xsd:sequence>
          <xsd:element name="specName" type="xsd:string"/>
          <xsd:element name="spec" type="ale:CCSpec"/>
        </xsd:sequence>
      </xsd:complexType>
      <xsd:element name="DefineResult">
        <xsd:complexType/>
      </xsd:element>

      <xsd:element name="Undefine" type="impl:Undefine"/>
      <xsd:complexType name="Undefine">
        <xsd:sequence>
          <xsd:element name="specName" type="xsd:string"/>
        </xsd:sequence>
      </xsd:complexType>
      <xsd:element name="UndefineResult">
        <xsd:complexType/>
      </xsd:element>

      <xsd:element name="GetCCSpec" type="impl:GetCCSpec"/>
      <xsd:complexType name="GetCCSpec">
        <xsd:sequence>
          <xsd:element name="specName" type="xsd:string"/>
        </xsd:sequence>
      </xsd:complexType>
      <xsd:element name="GetCCSpecResult" type="ale:CCSpec"/>

      <xsd:element name="GetCCSpecNames" type="impl:EmptyParms"/>
      <xsd:element name="GetCCSpecNamesResult" type="impl:ArrayOfString"/>

      <xsd:element name="Subscribe" type="impl:Subscribe"/>
      <xsd:complexType name="Subscribe">
        <xsd:sequence>
          <xsd:element name="specName" type="xsd:string"/>
          <xsd:element name="notificationURI" type="xsd:string"/>
        </xsd:sequence>
      </xsd:complexType>
      <xsd:element name="SubscribeResult">
        <xsd:complexType/>
      </xsd:element>

      <xsd:element name="Unsubscribe" type="impl:Unsubscribe"/>
      <xsd:complexType name="Unsubscribe">
        <xsd:sequence>
          <xsd:element name="specName" type="xsd:string"/>
          <xsd:element name="notificationURI" type="xsd:string"/>
        </xsd:sequence>
      </xsd:complexType>
      <xsd:element name="UnsubscribeResult">
        <xsd:complexType/>
      </xsd:element>

      <xsd:element name="Poll" type="impl:Poll"/>
      <xsd:complexType name="Poll">
        <xsd:sequence>
          <xsd:element name="specName" type="xsd:string"/>
          <xsd:element name="params" type="ale:CCParameterList"/>
        </xsd:sequence>
      </xsd:complexType>
      <xsd:element name="PollResult" type="ale:CCReports"/>

      <xsd:element name="Immediate" type="impl:Immediate"/>
      <xsd:complexType name="Immediate">
        <xsd:sequence>
          <xsd:element name="spec" type="ale:CCSpec"/>
        </xsd:sequence>
      </xsd:complexType>
      <xsd:element name="ImmediateResult" type="ale:CCReports"/>

      <xsd:element name="GetSubscribers" type="impl:GetSubscribers"/>
      <xsd:complexType name="GetSubscribers">
        <xsd:sequence>
          <xsd:element name="specName" type="xsd:string"/>
        </xsd:sequence>
      </xsd:complexType>
      <xsd:element name="GetSubscribersResult" type="impl:ArrayOfString"/>

      <xsd:element name="GetStandardVersion" type="impl:EmptyParms"/>
      <xsd:element name="GetStandardVersionResult" type="xsd:string"/>

      <xsd:element name="GetVendorVersion" type="impl:EmptyParms"/>
      <xsd:element name="GetVendorVersionResult" type="xsd:string"/>

      <xsd:element name="DefineEPCCache" type="impl:DefineEPCCache"/>
      <xsd:complexType name="DefineEPCCache">
        <xsd:sequence>
          <xsd:element name="cacheName" type="xsd:string"/>
          <xsd:element name="spec" type="ale:EPCCacheSpec"/>
          <xsd:element name="replenishment" type="ale:EPCPatternList"/>
        </xsd:sequence>
      </xsd:complexType>
      <xsd:element name="DefineEPCCacheResult">
        <xsd:complexType/>
      </xsd:element>

      <xsd:element name="UndefineEPCCache" type="impl:UndefineEPCCache"/>
      <xsd:complexType name="UndefineEPCCache">
        <xsd:sequence>
          <xsd:element name="cacheName" type="xsd:string"/>
        </xsd:sequence>
      </xsd:complexType>
      <xsd:element name="UndefineEPCCacheResult" type="ale:EPCPatternList"/>

      <xsd:element name="GetEPCCache" type="impl:GetEPCCache"/>
      <xsd:complexType name="GetEPCCache">
        <xsd:sequence>
          <xsd:element name="cacheName" type="xsd:string"/>
        </xsd:sequence>
      </xsd:complexType>
      <xsd:element name="GetEPCCacheResult" type="ale:EPCCacheSpec"/>

      <xsd:element name="GetEPCCacheNames" type="impl:EmptyParms"/>
      <xsd:element name="GetEPCCacheNamesResult" type="impl:ArrayOfString"/>
```

```

        <xsd:element name="ReplenishEPCCache" type="impl:ReplenishEPCCache"/>
        <xsd:complexType name="ReplenishEPCCache">
          <xsd:sequence>
            <xsd:element name="cacheName" type="xsd:string"/>
            <xsd:element name="replenishment" type="ale:EPCPatternList"/>
          </xsd:sequence>
        </xsd:complexType>
        <xsd:element name="ReplenishEPCCacheResult">
          <xsd:complexType/>
        </xsd:element>

        <xsd:element name="DepleteEPCCache" type="impl:DepleteEPCCache"/>
        <xsd:complexType name="DepleteEPCCache">
          <xsd:sequence>
            <xsd:element name="cacheName" type="xsd:string"/>
          </xsd:sequence>
        </xsd:complexType>
        <xsd:element name="DepleteEPCCacheResult" type="ale:EPCPatternList"/>

        <xsd:element name="GetEPCCacheContents" type="impl:GetEPCCacheContents"/>
        <xsd:complexType name="GetEPCCacheContents">
          <xsd:sequence>
            <xsd:element name="cacheName" type="xsd:string"/>
          </xsd:sequence>
        </xsd:complexType>
        <xsd:element name="GetEPCCacheContentsResult" type="ale:EPCPatternList"/>

        <xsd:element name="DefineAssocTable" type="impl:DefineAssocTable"/>
        <xsd:complexType name="DefineAssocTable">
          <xsd:sequence>
            <xsd:element name="tableName" type="xsd:string"/>
            <xsd:element name="spec" type="ale:AssocTableSpec"/>
            <xsd:element name="entries" type="ale:AssocTableEntryList"/>
          </xsd:sequence>
        </xsd:complexType>
        <xsd:element name="DefineAssocTableResult">
          <xsd:complexType/>
        </xsd:element>

        <xsd:element name="UndefineAssocTable" type="impl:UndefineAssocTable"/>
        <xsd:complexType name="UndefineAssocTable">
          <xsd:sequence>
            <xsd:element name="tableName" type="xsd:string"/>
          </xsd:sequence>
        </xsd:complexType>
        <xsd:element name="UndefineAssocTableResult">
          <xsd:complexType/>
        </xsd:element>

        <xsd:element name="GetAssocTableNames" type="impl:EmptyParms"/>
        <xsd:element name="GetAssocTableNamesResult" type="impl:ArrayOfString"/>

        <xsd:element name="GetAssocTable" type="impl:GetAssocTable"/>
        <xsd:complexType name="GetAssocTable">
          <xsd:sequence>
            <xsd:element name="tableName" type="xsd:string"/>
          </xsd:sequence>
        </xsd:complexType>
        <xsd:element name="GetAssocTableResult" type="ale:AssocTableSpec"/>

        <xsd:element name="PutAssocTableEntries" type="impl:PutAssocTableEntries"/>
        <xsd:complexType name="PutAssocTableEntries">
          <xsd:sequence>
            <xsd:element name="tableName" type="xsd:string"/>
            <xsd:element name="entries" type="ale:AssocTableEntryList"/>
          </xsd:sequence>
        </xsd:complexType>
        <xsd:element name="PutAssocTableEntriesResult">
          <xsd:complexType/>
        </xsd:element>

        <xsd:element name="GetAssocTableValue" type="impl:GetAssocTableValue"/>
        <xsd:complexType name="GetAssocTableValue">
          <xsd:sequence>
            <xsd:element name="tableName" type="xsd:string"/>
            <xsd:element name="epc" type="xsd:string"/>
          </xsd:sequence>
        </xsd:complexType>
        <xsd:element name="GetAssocTableValueResult" type="xsd:string"/>

        <xsd:element name="GetAssocTableEntries" type="impl:GetAssocTableEntries"/>
        <xsd:complexType name="GetAssocTableEntries">
          <xsd:sequence>
            <xsd:element name="tableName" type="xsd:string"/>
            <xsd:element name="patList" type="ale:EPCPatternList"/>
          </xsd:sequence>
        </xsd:complexType>
        <xsd:element name="GetAssocTableEntriesResult" type="ale:AssocTableEntryList"/>

        <xsd:element name="RemoveAssocTableEntry" type="impl:RemoveAssocTableEntry"/>
        <xsd:complexType name="RemoveAssocTableEntry">
          <xsd:sequence>
            <xsd:element name="tableName" type="xsd:string"/>
            <xsd:element name="epc" type="xsd:string"/>
          </xsd:sequence>
        </xsd:complexType>
        <xsd:element name="RemoveAssocTableEntryResult">
          <xsd:complexType/>
        </xsd:element>

        <xsd:element name="RemoveAssocTableEntries" type="impl:RemoveAssocTableEntries"/>
        <xsd:complexType name="RemoveAssocTableEntries">
          <xsd:sequence>
            <xsd:element name="tableName" type="xsd:string"/>
            <xsd:element name="patList" type="ale:EPCPatternList"/>
          </xsd:sequence>
        </xsd:complexType>
        <xsd:element name="RemoveAssocTableEntriesResult">
          <xsd:complexType/>
        </xsd:element>

        <xsd:element name="DefineRNG" type="impl:DefineRNG"/>
        <xsd:complexType name="DefineRNG">
          <xsd:sequence>
            <xsd:element name="rngName" type="xsd:string"/>
            <xsd:element name="rngSpec" type="ale:RNGSpec"/>
          </xsd:sequence>
        </xsd:complexType>
        <xsd:element name="DefineRNGResult">
          <xsd:complexType/>
        </xsd:element>

        <xsd:element name="UndefineRNG" type="impl:UndefineRNG"/>
        <xsd:complexType name="UndefineRNG">
          <xsd:sequence>
            <xsd:element name="rngName" type="xsd:string"/>
          </xsd:sequence>
        </xsd:complexType>
        <xsd:element name="UndefineRNGResult">
          <xsd:complexType/>
        </xsd:element>

        <xsd:element name="GetRNGNames" type="impl:EmptyParms"/>
        <xsd:element name="GetRNGNamesResult" type="impl:ArrayOfString"/>

        <xsd:element name="GetRNG" type="impl:GetRNG"/>
        <xsd:complexType name="GetRNG">
          <xsd:sequence>
            <xsd:element name="rngName" type="xsd:string"/>
```

```
    </xsd:sequence>
  </xsd:complexType>
  <xsd:element name="GetRNGResult" type="ale:RNGSpec"/>

  <xsd:element name="ALEException" type="impl:ALEException"/>
  <xsd:complexType name="ALEException">
    <xsd:sequence>
      <xsd:element name="reason" type="xsd:string"/>
    </xsd:sequence>
  </xsd:complexType>

  <xsd:element name="SecurityException" type="impl:SecurityException"/>
  <xsd:complexType name="SecurityException">
    <xsd:complexContent>
      <xsd:extension base="impl:ALEException"/>
    </xsd:complexContent>
  </xsd:complexType>

  <xsd:element name="DuplicateNameException" type="impl:DuplicateNameException"/>
  <xsd:complexType name="DuplicateNameException">
    <xsd:complexContent>
      <xsd:extension base="impl:ALEException"/>
    </xsd:complexContent>
  </xsd:complexType>

  <xsd:element name="CCSpecValidationException"
               type="impl:CCSpecValidationException"/>
  <xsd:complexType name="CCSpecValidationException">
    <xsd:complexContent>
      <xsd:extension base="impl:ALEException"/>
    </xsd:complexContent>
  </xsd:complexType>

  <xsd:element name="InvalidURIException" type="impl:InvalidURIException"/>
  <xsd:complexType name="InvalidURIException">
    <xsd:complexContent>
      <xsd:extension base="impl:ALEException"/>
    </xsd:complexContent>
  </xsd:complexType>

  <xsd:element name="NoSuchNameException" type="impl:NoSuchNameException"/>
  <xsd:complexType name="NoSuchNameException">
    <xsd:complexContent>
      <xsd:extension base="impl:ALEException"/>
    </xsd:complexContent>
  </xsd:complexType>

  <xsd:element name="NoSuchSubscriberException"
               type="impl:NoSuchSubscriberException"/>
  <xsd:complexType name="NoSuchSubscriberException">
    <xsd:complexContent>
      <xsd:extension base="impl:ALEException"/>
    </xsd:complexContent>
  </xsd:complexType>

  <xsd:element name="DuplicateSubscriptionException"
               type="impl:DuplicateSubscriptionException"/>
  <xsd:complexType name="DuplicateSubscriptionException">
    <xsd:complexContent>
      <xsd:extension base="impl:ALEException"/>
    </xsd:complexContent>
  </xsd:complexType>

  <xsd:element name="ParameterException" type="impl:ParameterException"/>
  <xsd:complexType name="ParameterException">
    <xsd:complexContent>
      <xsd:extension base="impl:ALEException"/>
    </xsd:complexContent>
  </xsd:complexType>

```

```
        <xsd:element name="ParameterForbiddenException"
                     type="impl:ParameterForbiddenException"/>
        <xsd:complexType name="ParameterForbiddenException">
          <xsd:complexContent>
            <xsd:extension base="impl:ALEException"/>
          </xsd:complexContent>
        </xsd:complexType>

        <xsd:element name="ImplementationException" type="impl:ImplementationException"/>
        <xsd:complexType name="ImplementationException">
          <xsd:complexContent>
            <xsd:extension base="impl:ALEException">
              <xsd:sequence>
                <xsd:element name="severity" type="impl:ImplementationExceptionSeverity"/>
              </xsd:sequence>
            </xsd:extension>
          </xsd:complexContent>
        </xsd:complexType>

        <xsd:element name="EPCCacheSpecValidationException"
                     type="impl:EPCCacheSpecValidationException"/>
        <xsd:complexType name="EPCCacheSpecValidationException">
          <xsd:complexContent>
            <xsd:extension base="impl:ALEException"/>
          </xsd:complexContent>
        </xsd:complexType>

        <xsd:element name="InvalidPatternException" type="impl:InvalidPatternException"/>
        <xsd:complexType name="InvalidPatternException">
          <xsd:complexContent>
            <xsd:extension base="impl:ALEException"/>
          </xsd:complexContent>
        </xsd:complexType>

        <xsd:element name="InUseException" type="impl:InUseException"/>
        <xsd:complexType name="InUseException">
          <xsd:complexContent>
            <xsd:extension base="impl:ALEException"/>
          </xsd:complexContent>
        </xsd:complexType>

        <xsd:element name="AssocTableValidationException"
                     type="impl:AssocTableValidationException"/>
        <xsd:complexType name="AssocTableValidationException">
          <xsd:complexContent>
            <xsd:extension base="impl:ALEException"/>
          </xsd:complexContent>
        </xsd:complexType>

        <xsd:element name="InvalidEPCException" type="impl:InvalidEPCException"/>
        <xsd:complexType name="InvalidEPCException">
          <xsd:complexContent>
            <xsd:extension base="impl:ALEException"/>
          </xsd:complexContent>
        </xsd:complexType>

        <xsd:element name="InvalidAssocTableEntryException"
                     type="impl:InvalidAssocTableEntryException"/>
        <xsd:complexType name="InvalidAssocTableEntryException">
          <xsd:complexContent>
            <xsd:extension base="impl:ALEException"/>
          </xsd:complexContent>
        </xsd:complexType>

        <xsd:element name="RNGValidationException" type="impl:RNGValidationException"/>
        <xsd:complexType name="RNGValidationException">
          <xsd:complexContent>
            <xsd:extension base="impl:ALEException"/>
          </xsd:complexContent>
        </xsd:complexType>
```

```

    <xsd:complexType name="ArrayOfString">
      <xsd:sequence>
        <xsd:element name="string" type="xsd:string" minOccurs="0"
                     maxOccurs="unbounded"/>
      </xsd:sequence>
    </xsd:complexType>

    <!-- The ImplementationExceptionSeverity type is an enumerated type.
         The following strings are legal values for this type:
           ERROR
           SEVERE
          -->
    <xsd:simpleType name="ImplementationExceptionSeverity">
      <xsd:restriction base="xsd:string"/>
    </xsd:simpleType>

    <xsd:complexType name="EmptyParms"/>
  </xsd:schema>
</wsdl:types>
<!-- ALECCSERVICE MESSAGES -->

<wsdl:message name="defineRequest">
  <wsdl:part name="parms" element="impl:Define"/>
</wsdl:message>
<wsdl:message name="defineResponse">
  <wsdl:part name="defineReturn" element="impl:DefineResult"/>
</wsdl:message>

<wsdl:message name="undefineRequest">
  <wsdl:part name="parms" element="impl:Undefine"/>
</wsdl:message>
<wsdl:message name="undefineResponse">
  <wsdl:part name="undefineReturn" element="impl:UndefineResult"/>
</wsdl:message>

<wsdl:message name="getCCSpecRequest">
  <wsdl:part name="parms" element="impl:GetCCSpec"/>
</wsdl:message>
<wsdl:message name="getCCSpecResponse">
  <wsdl:part name="getCCSpecReturn" element="impl:GetCCSpecResult"/>
</wsdl:message>

<wsdl:message name="getCCSpecNamesRequest">
  <wsdl:part name="parms" element="impl:GetCCSpecNames"/>
</wsdl:message>
<wsdl:message name="getCCSpecNamesResponse">
  <wsdl:part name="getCCSpecNamesReturn" element="impl:GetCCSpecNamesResult"/>
</wsdl:message>

<wsdl:message name="subscribeRequest">
  <wsdl:part name="parms" element="impl:Subscribe"/>
</wsdl:message>
<wsdl:message name="subscribeResponse">
  <wsdl:part name="subscribeReturn" element="impl:SubscribeResult"/>
</wsdl:message>

<wsdl:message name="unsubscribeRequest">
  <wsdl:part name="parms" element="impl:Unsubscribe"/>
</wsdl:message>
<wsdl:message name="unsubscribeResponse">
  <wsdl:part name="unsubscribeReturn" element="impl:UnsubscribeResult"/>
</wsdl:message>

<wsdl:message name="pollRequest">
  <wsdl:part name="parms" element="impl:Poll"/>
</wsdl:message>
<wsdl:message name="pollResponse">
  <wsdl:part name="pollReturn" element="impl:PollResult"/>
</wsdl:message>
```

```
  <wsdl:message name="immediateRequest">
    <wsdl:part name="parms" element="impl:Immediate"/>
  </wsdl:message>
  <wsdl:message name="immediateResponse">
    <wsdl:part name="immediateReturn" element="impl:ImmediateResult"/>
  </wsdl:message>

  <wsdl:message name="getSubscribersRequest">
    <wsdl:part name="parms" element="impl:GetSubscribers"/>
  </wsdl:message>
  <wsdl:message name="getSubscribersResponse">
    <wsdl:part name="getSubscribersReturn" element="impl:GetSubscribersResult"/>
  </wsdl:message>

  <wsdl:message name="getStandardVersionRequest">
    <wsdl:part name="parms" element="impl:GetStandardVersion"/>
  </wsdl:message>
  <wsdl:message name="getStandardVersionResponse">
    <wsdl:part name="getStandardVersionReturn" element="impl:GetStandardVersionResult"/>
  </wsdl:message>

  <wsdl:message name="getVendorVersionRequest">
    <wsdl:part name="parms" element="impl:GetVendorVersion"/>
  </wsdl:message>
  <wsdl:message name="getVendorVersionResponse">
    <wsdl:part name="getVendorVersionReturn" element="impl:GetVendorVersionResult"/>
  </wsdl:message>

  <wsdl:message name="defineEPCCacheRequest">
    <wsdl:part name="parms" element="impl:DefineEPCCache"/>
  </wsdl:message>
  <wsdl:message name="defineEPCCacheResponse">
    <wsdl:part name="defineEPCCacheReturn" element="impl:DefineEPCCacheResult"/>
  </wsdl:message>

  <wsdl:message name="undefineEPCCacheRequest">
    <wsdl:part name="parms" element="impl:UndefineEPCCache"/>
  </wsdl:message>
  <wsdl:message name="undefineEPCCacheResponse">
    <wsdl:part name="undefineEPCCacheReturn" element="impl:UndefineEPCCacheResult"/>
  </wsdl:message>

  <wsdl:message name="getEPCCacheRequest">
    <wsdl:part name="parms" element="impl:GetEPCCache"/>
  </wsdl:message>
  <wsdl:message name="getEPCCacheResponse">
    <wsdl:part name="getEPCCacheReturn" element="impl:GetEPCCacheResult"/>
  </wsdl:message>

  <wsdl:message name="getEPCCacheNamesRequest">
    <wsdl:part name="parms" element="impl:GetEPCCacheNames"/>
  </wsdl:message>
  <wsdl:message name="getEPCCacheNamesResponse">
    <wsdl:part name="getEPCCacheNamesReturn" element="impl:GetEPCCacheNamesResult"/>
  </wsdl:message>

  <wsdl:message name="replenishEPCCacheRequest">
    <wsdl:part name="parms" element="impl:ReplenishEPCCache"/>
  </wsdl:message>
  <wsdl:message name="replenishEPCCacheResponse">
    <wsdl:part name="replenishEPCCacheReturn" element="impl:ReplenishEPCCacheResult"/>
  </wsdl:message>

  <wsdl:message name="depleteEPCCacheRequest">
    <wsdl:part name="parms" element="impl:DepleteEPCCache"/>
  </wsdl:message>
  <wsdl:message name="depleteEPCCacheResponse">
    <wsdl:part name="depleteEPCCacheReturn" element="impl:DepleteEPCCacheResult"/>
  </wsdl:message>

  <wsdl:message name="getEPCCacheContentsRequest">
    <wsdl:part name="parms" element="impl:GetEPCCacheContents"/>
  </wsdl:message>
  <wsdl:message name="getEPCCacheContentsResponse">
    <wsdl:part name="getEPCCacheContentsReturn"
               element="impl:GetEPCCacheContentsResult"/>
  </wsdl:message>

  <wsdl:message name="defineAssocTableRequest">
    <wsdl:part name="parms" element="impl:DefineAssocTable"/>
  </wsdl:message>
  <wsdl:message name="defineAssocTableResponse">
    <wsdl:part name="defineAssocTableReturn" element="impl:DefineAssocTableResult"/>
  </wsdl:message>

  <wsdl:message name="undefineAssocTableRequest">
    <wsdl:part name="parms" element="impl:UndefineAssocTable"/>
  </wsdl:message>
  <wsdl:message name="undefineAssocTableResponse">
    <wsdl:part name="undefineAssocTableReturn" element="impl:UndefineAssocTableResult"/>
  </wsdl:message>

  <wsdl:message name="getAssocTableNamesRequest">
    <wsdl:part name="parms" element="impl:GetAssocTableNames"/>
  </wsdl:message>
  <wsdl:message name="getAssocTableNamesResponse">
    <wsdl:part name="getAssocTableNamesReturn" element="impl:GetAssocTableNamesResult"/>
  </wsdl:message>

  <wsdl:message name="getAssocTableRequest">
    <wsdl:part name="parms" element="impl:GetAssocTable"/>
  </wsdl:message>
  <wsdl:message name="getAssocTableResponse">
    <wsdl:part name="getAssocTableReturn" element="impl:GetAssocTableResult"/>
  </wsdl:message>

  <wsdl:message name="putAssocTableEntriesRequest">
    <wsdl:part name="parms" element="impl:PutAssocTableEntries"/>
  </wsdl:message>
  <wsdl:message name="putAssocTableEntriesResponse">
    <wsdl:part name="putAssocTableEntriesReturn"
               element="impl:PutAssocTableEntriesResult"/>
  </wsdl:message>

  <wsdl:message name="getAssocTableValueRequest">
    <wsdl:part name="parms" element="impl:GetAssocTableValue"/>
  </wsdl:message>
  <wsdl:message name="getAssocTableValueResponse">
    <wsdl:part name="getAssocTableValueReturn" element="impl:GetAssocTableValueResult"/>
  </wsdl:message>

  <wsdl:message name="getAssocTableEntriesRequest">
    <wsdl:part name="parms" element="impl:GetAssocTableEntries"/>
  </wsdl:message>
  <wsdl:message name="getAssocTableEntriesResponse">
    <wsdl:part name="getAssocTableEntriesReturn"
               element="impl:GetAssocTableEntriesResult"/>
  </wsdl:message>

  <wsdl:message name="removeAssocTableEntryRequest">
    <wsdl:part name="parms" element="impl:RemoveAssocTableEntry"/>
  </wsdl:message>
  <wsdl:message name="removeAssocTableEntryResponse">
    <wsdl:part name="removeAssocTableEntryReturn"
               element="impl:RemoveAssocTableEntryResult"/>
  </wsdl:message>

  <wsdl:message name="removeAssocTableEntriesRequest">
    <wsdl:part name="parms" element="impl:RemoveAssocTableEntries"/>
```

```
</wsdl:message>
<wsdl:message name="removeAssocTableEntriesResponse">
  <wsdl:part name="removeAssocTableEntriesReturn"
             element="impl:RemoveAssocTableEntriesResult"/>
</wsdl:message>

<wsdl:message name="defineRNGRequest">
  <wsdl:part name="parms" element="impl:DefineRNG"/>
</wsdl:message>
<wsdl:message name="defineRNGResponse">
  <wsdl:part name="defineRNGReturn" element="impl:DefineRNGResult"/>
</wsdl:message>

<wsdl:message name="undefineRNGRequest">
  <wsdl:part name="parms" element="impl:UndefineRNG"/>
</wsdl:message>
<wsdl:message name="undefineRNGResponse">
  <wsdl:part name="undefineRNGReturn" element="impl:UndefineRNGResult"/>
</wsdl:message>

<wsdl:message name="getRNGNamesRequest">
  <wsdl:part name="parms" element="impl:GetRNGNames"/>
</wsdl:message>
<wsdl:message name="getRNGNamesResponse">
  <wsdl:part name="getRNGNamesReturn" element="impl:GetRNGNamesResult"/>
</wsdl:message>

<wsdl:message name="getRNGRequest">
  <wsdl:part name="parms" element="impl:GetRNG"/>
</wsdl:message>
<wsdl:message name="getRNGResponse">
  <wsdl:part name="getRNGReturn" element="impl:GetRNGResult"/>
</wsdl:message>

<wsdl:message name="SecurityExceptionResponse">
  <wsdl:part name="fault" element="impl:SecurityException"/>
</wsdl:message>

<wsdl:message name="DuplicateNameExceptionResponse">
  <wsdl:part name="fault" element="impl:DuplicateNameException"/>
</wsdl:message>

<wsdl:message name="CCSpecValidationExceptionResponse">
  <wsdl:part name="fault" element="impl:CCSpecValidationException"/>
</wsdl:message>

<wsdl:message name="InvalidURIExceptionResponse">
  <wsdl:part name="fault" element="impl:InvalidURIException"/>
</wsdl:message>

<wsdl:message name="NoSuchNameExceptionResponse">
  <wsdl:part name="fault" element="impl:NoSuchNameException"/>
</wsdl:message>

<wsdl:message name="NoSuchSubscriberExceptionResponse">
  <wsdl:part name="fault" element="impl:NoSuchSubscriberException"/>
</wsdl:message>

<wsdl:message name="DuplicateSubscriptionExceptionResponse">
  <wsdl:part name="fault" element="impl:DuplicateSubscriptionException"/>
</wsdl:message>

<wsdl:message name="ParameterExceptionResponse">
  <wsdl:part name="fault" element="impl:ParameterException"/>
</wsdl:message>

<wsdl:message name="ParameterForbiddenExceptionResponse">
  <wsdl:part name="fault" element="impl:ParameterForbiddenException"/>
</wsdl:message>

```

```
<wsdl:message name="ImplementationExceptionResponse">
  <wsdl:part name="fault" element="impl:ImplementationException"/>
</wsdl:message>

<wsdl:message name="EPCCacheSpecValidationExceptionResponse">
  <wsdl:part name="fault" element="impl:EPCCacheSpecValidationException"/>
</wsdl:message>

<wsdl:message name="InvalidPatternExceptionResponse">
  <wsdl:part name="fault" element="impl:InvalidPatternException"/>
</wsdl:message>

<wsdl:message name="InUseExceptionResponse">
  <wsdl:part name="fault" element="impl:InUseException"/>
</wsdl:message>

<wsdl:message name="AssocTableValidationExceptionResponse">
  <wsdl:part name="fault" element="impl:AssocTableValidationException"/>
</wsdl:message>

<wsdl:message name="InvalidEPCExceptionResponse">
  <wsdl:part name="fault" element="impl:InvalidEPCException"/>
</wsdl:message>

<wsdl:message name="InvalidAssocTableEntryExceptionResponse">
  <wsdl:part name="fault" element="impl:InvalidAssocTableEntryException"/>
</wsdl:message>

<wsdl:message name="RNGValidationExceptionResponse">
  <wsdl:part name="fault" element="impl:RNGValidationException"/>
</wsdl:message>
<!-- ALECCSERVICE PORTTYPE -->

<wsdl:portType name="ALECCServicePortType">

  <wsdl:operation name="define">
    <wsdl:input message="impl:defineRequest" name="defineRequest"/>
    <wsdl:output message="impl:defineResponse" name="defineResponse"/>
    <wsdl:fault message="impl:DuplicateNameExceptionResponse"
                name="DuplicateNameExceptionFault"/>
    <wsdl:fault message="impl:CCSpecValidationExceptionResponse"
                name="CCSpecValidationExceptionFault"/>
    <wsdl:fault message="impl:SecurityExceptionResponse"
                name="SecurityExceptionFault"/>
    <wsdl:fault message="impl:ImplementationExceptionResponse"
                name="ImplementationExceptionFault"/>
  </wsdl:operation>

  <wsdl:operation name="undefine">
    <wsdl:input message="impl:undefineRequest" name="undefineRequest"/>
    <wsdl:output message="impl:undefineResponse" name="undefineResponse"/>
    <wsdl:fault message="impl:NoSuchNameExceptionResponse"
                name="NoSuchNameExceptionFault"/>
    <wsdl:fault message="impl:SecurityExceptionResponse"
                name="SecurityExceptionFault"/>
    <wsdl:fault message="impl:ImplementationExceptionResponse"
                name="ImplementationExceptionFault"/>
  </wsdl:operation>

  <wsdl:operation name="getCCSpec">
    <wsdl:input message="impl:getCCSpecRequest" name="getCCSpecRequest"/>
    <wsdl:output message="impl:getCCSpecResponse" name="getCCSpecResponse"/>
    <wsdl:fault message="impl:NoSuchNameExceptionResponse"
                name="NoSuchNameExceptionFault"/>
    <wsdl:fault message="impl:SecurityExceptionResponse"
                name="SecurityExceptionFault"/>
    <wsdl:fault message="impl:ImplementationExceptionResponse"
                name="ImplementationExceptionFault"/>
  </wsdl:operation>

```

```
      <wsdl:operation name="getCCSpecNames">
        <wsdl:input message="impl:getCCSpecNamesRequest" name="getCCSpecNamesRequest"/>
        <wsdl:output message="impl:getCCSpecNamesResponse" name="getCCSpecNamesResponse"/>
        <wsdl:fault message="impl:SecurityExceptionResponse"
                    name="SecurityExceptionFault"/>
        <wsdl:fault message="impl:ImplementationExceptionResponse"
                    name="ImplementationExceptionFault"/>
      </wsdl:operation>

      <wsdl:operation name="subscribe">
        <wsdl:input message="impl:subscribeRequest" name="subscribeRequest"/>
        <wsdl:output message="impl:subscribeResponse" name="subscribeResponse"/>
        <wsdl:fault message="impl:NoSuchNameExceptionResponse"
                    name="NoSuchNameExceptionFault"/>
        <wsdl:fault message="impl:InvalidURIExceptionResponse"
                    name="InvalidURIExceptionFault"/>
        <wsdl:fault message="impl:DuplicateSubscriptionExceptionResponse"
                    name="DuplicateSubscriptionExceptionFault"/>
        <wsdl:fault message="impl:ParameterForbiddenExceptionResponse"
                    name="ParameterForbiddenExceptionFault"/>
        <wsdl:fault message="impl:SecurityExceptionResponse"
                    name="SecurityExceptionFault"/>
        <wsdl:fault message="impl:ImplementationExceptionResponse"
                    name="ImplementationExceptionFault"/>
      </wsdl:operation>

      <wsdl:operation name="unsubscribe">
        <wsdl:input message="impl:unsubscribeRequest" name="unsubscribeRequest"/>
        <wsdl:output message="impl:unsubscribeResponse" name="unsubscribeResponse"/>
        <wsdl:fault message="impl:NoSuchNameExceptionResponse"
                    name="NoSuchNameExceptionFault"/>
        <wsdl:fault message="impl:NoSuchSubscriberExceptionResponse"
                    name="NoSuchSubscriberExceptionFault"/>
        <wsdl:fault message="impl:InvalidURIExceptionResponse"
                    name="InvalidURIExceptionFault"/>
        <wsdl:fault message="impl:SecurityExceptionResponse"
                    name="SecurityExceptionFault"/>
        <wsdl:fault message="impl:ImplementationExceptionResponse"
                    name="ImplementationExceptionFault"/>
      </wsdl:operation>

      <wsdl:operation name="poll">
        <wsdl:input message="impl:pollRequest" name="pollRequest"/>
        <wsdl:output message="impl:pollResponse" name="pollResponse"/>
        <wsdl:fault message="impl:NoSuchNameExceptionResponse"
                    name="NoSuchNameExceptionFault"/>
        <wsdl:fault message="impl:ParameterExceptionResponse"
                    name="ParameterExceptionFault"/>
        <wsdl:fault message="impl:SecurityExceptionResponse"
                    name="SecurityExceptionFault"/>
        <wsdl:fault message="impl:ImplementationExceptionResponse"
                    name="ImplementationExceptionFault"/>
      </wsdl:operation>

      <wsdl:operation name="immediate">
        <wsdl:input message="impl:immediateRequest" name="immediateRequest"/>
        <wsdl:output message="impl:immediateResponse" name="immediateResponse"/>
        <wsdl:fault message="impl:CCSpecValidationExceptionResponse"
                    name="CCSpecValidationExceptionFault"/>
        <wsdl:fault message="impl:ParameterForbiddenExceptionResponse"
                    name="ParameterForbiddenExceptionFault"/>
        <wsdl:fault message="impl:SecurityExceptionResponse"
                    name="SecurityExceptionFault"/>
        <wsdl:fault message="impl:ImplementationExceptionResponse"
                    name="ImplementationExceptionFault"/>
      </wsdl:operation>

      <wsdl:operation name="getSubscribers">
        <wsdl:input message="impl:getSubscribersRequest" name="getSubscribersRequest"/>
        <wsdl:output message="impl:getSubscribersResponse" name="getSubscribersResponse"/>
```

```
    <wsdl:fault message="impl:NoSuchNameExceptionResponse"
                name="NoSuchNameExceptionFault"/>
    <wsdl:fault message="impl:SecurityExceptionResponse"
                name="SecurityExceptionFault"/>
    <wsdl:fault message="impl:ImplementationExceptionResponse"
                name="ImplementationExceptionFault"/>
  </wsdl:operation>

  <wsdl:operation name="getStandardVersion">
    <wsdl:input message="impl:getStandardVersionRequest"
                name="getStandardVersionRequest"/>
    <wsdl:output message="impl:getStandardVersionResponse"
                 name="getStandardVersionResponse"/>
    <wsdl:fault message="impl:ImplementationExceptionResponse"
                name="ImplementationExceptionFault"/>
  </wsdl:operation>

  <wsdl:operation name="getVendorVersion">
    <wsdl:input message="impl:getVendorVersionRequest" name="getVendorVersionRequest"/>
    <wsdl:output message="impl:getVendorVersionResponse"
                 name="getVendorVersionResponse"/>
    <wsdl:fault message="impl:ImplementationExceptionResponse"
                name="ImplementationExceptionFault"/>
  </wsdl:operation>

  <wsdl:operation name="defineEPCCache">
    <wsdl:input message="impl:defineEPCCacheRequest" name="defineEPCCacheRequest"/>
    <wsdl:output message="impl:defineEPCCacheResponse" name="defineEPCCacheResponse"/>
    <wsdl:fault message="impl:DuplicateNameExceptionResponse"
                name="DuplicateNameExceptionFault"/>
    <wsdl:fault message="impl:EPCCacheSpecValidationExceptionResponse"
                name="EPCCacheSpecValidationExceptionFault"/>
    <wsdl:fault message="impl:InvalidPatternExceptionResponse"
                name="InvalidPatternExceptionFault"/>
    <wsdl:fault message="impl:SecurityExceptionResponse"
                name="SecurityExceptionFault"/>
    <wsdl:fault message="impl:ImplementationExceptionResponse"
                name="ImplementationExceptionFault"/>
  </wsdl:operation>

  <wsdl:operation name="undefineEPCCache">
    <wsdl:input message="impl:undefineEPCCacheRequest" name="undefineEPCCacheRequest"/>
    <wsdl:output message="impl:undefineEPCCacheResponse"
                 name="undefineEPCCacheResponse"/>
    <wsdl:fault message="impl:NoSuchNameExceptionResponse"
                name="NoSuchNameExceptionFault"/>
    <wsdl:fault message="impl:InUseExceptionResponse" name="InUseExceptionFault"/>
    <wsdl:fault message="impl:SecurityExceptionResponse"
                name="SecurityExceptionFault"/>
    <wsdl:fault message="impl:ImplementationExceptionResponse"
                name="ImplementationExceptionFault"/>
  </wsdl:operation>

  <wsdl:operation name="getEPCCache">
    <wsdl:input message="impl:getEPCCacheRequest" name="getEPCCacheRequest"/>
    <wsdl:output message="impl:getEPCCacheResponse" name="getEPCCacheResponse"/>
    <wsdl:fault message="impl:NoSuchNameExceptionResponse"
                name="NoSuchNameExceptionFault"/>
    <wsdl:fault message="impl:SecurityExceptionResponse"
                name="SecurityExceptionFault"/>
    <wsdl:fault message="impl:ImplementationExceptionResponse"
                name="ImplementationExceptionFault"/>
  </wsdl:operation>

  <wsdl:operation name="getEPCCacheNames">
    <wsdl:input message="impl:getEPCCacheNamesRequest" name="getEPCCacheNamesRequest"/>
    <wsdl:output message="impl:getEPCCacheNamesResponse"
                 name="getEPCCacheNamesResponse"/>
    <wsdl:fault message="impl:SecurityExceptionResponse"
                name="SecurityExceptionFault"/>
```

```
      <wsdl:fault message="impl:ImplementationExceptionResponse"
                  name="ImplementationExceptionFault"/>
    </wsdl:operation>

    <wsdl:operation name="replenishEPCCache">
      <wsdl:input message="impl:replenishEPCCacheRequest"
                  name="replenishEPCCacheRequest"/>
      <wsdl:output message="impl:replenishEPCCacheResponse"
                   name="replenishEPCCacheResponse"/>
      <wsdl:fault message="impl:NoSuchNameExceptionResponse"
                  name="NoSuchNameExceptionFault"/>
      <wsdl:fault message="impl:InvalidPatternExceptionResponse"
                  name="InvalidPatternExceptionFault"/>
      <wsdl:fault message="impl:SecurityExceptionResponse"
                  name="SecurityExceptionFault"/>
      <wsdl:fault message="impl:ImplementationExceptionResponse"
                  name="ImplementationExceptionFault"/>
    </wsdl:operation>

    <wsdl:operation name="depleteEPCCache">
      <wsdl:input message="impl:depleteEPCCacheRequest" name="depleteEPCCacheRequest"/>
      <wsdl:output message="impl:depleteEPCCacheResponse"
                   name="depleteEPCCacheResponse"/>
      <wsdl:fault message="impl:NoSuchNameExceptionResponse"
                  name="NoSuchNameExceptionFault"/>
      <wsdl:fault message="impl:SecurityExceptionResponse"
                  name="SecurityExceptionFault"/>
      <wsdl:fault message="impl:ImplementationExceptionResponse"
                  name="ImplementationExceptionFault"/>
    </wsdl:operation>

    <wsdl:operation name="getEPCCacheContents">
      <wsdl:input message="impl:getEPCCacheContentsRequest"
                  name="getEPCCacheContentsRequest"/>
      <wsdl:output message="impl:getEPCCacheContentsResponse"
                   name="getEPCCacheContentsResponse"/>
      <wsdl:fault message="impl:NoSuchNameExceptionResponse"
                  name="NoSuchNameExceptionFault"/>
      <wsdl:fault message="impl:SecurityExceptionResponse"
                  name="SecurityExceptionFault"/>
      <wsdl:fault message="impl:ImplementationExceptionResponse"
                  name="ImplementationExceptionFault"/>
    </wsdl:operation>

    <wsdl:operation name="defineAssocTable">
      <wsdl:input message="impl:defineAssocTableRequest" name="defineAssocTableRequest"/>
      <wsdl:output message="impl:defineAssocTableResponse"
                   name="defineAssocTableResponse"/>
      <wsdl:fault message="impl:DuplicateNameExceptionResponse"
                  name="DuplicateNameExceptionFault"/>
      <wsdl:fault message="impl:AssocTableValidationExceptionResponse"
                  name="AssocTableValidationExceptionFault"/>
      <wsdl:fault message="impl:InvalidAssocTableEntryExceptionResponse"
                  name="InvalidAssocTableEntryExceptionFault"/>
      <wsdl:fault message="impl:SecurityExceptionResponse"
                  name="SecurityExceptionFault"/>
      <wsdl:fault message="impl:ImplementationExceptionResponse"
                  name="ImplementationExceptionFault"/>
    </wsdl:operation>

    <wsdl:operation name="undefineAssocTable">
      <wsdl:input message="impl:undefineAssocTableRequest"
                  name="undefineAssocTableRequest"/>
      <wsdl:output message="impl:undefineAssocTableResponse"
                   name="undefineAssocTableResponse"/>
      <wsdl:fault message="impl:NoSuchNameExceptionResponse"
                  name="NoSuchNameExceptionFault"/>
      <wsdl:fault message="impl:InUseExceptionResponse" name="InUseExceptionFault"/>
      <wsdl:fault message="impl:SecurityExceptionResponse"
                  name="SecurityExceptionFault"/>
      <wsdl:fault message="impl:ImplementationExceptionResponse"
                  name="ImplementationExceptionFault"/>
    </wsdl:operation>

    <wsdl:operation name="getAssocTableNames">
      <wsdl:input message="impl:getAssocTableNamesRequest"
                  name="getAssocTableNamesRequest"/>
      <wsdl:output message="impl:getAssocTableNamesResponse"
                   name="getAssocTableNamesResponse"/>
      <wsdl:fault message="impl:SecurityExceptionResponse"
                  name="SecurityExceptionFault"/>
      <wsdl:fault message="impl:ImplementationExceptionResponse"
                  name="ImplementationExceptionFault"/>
    </wsdl:operation>

    <wsdl:operation name="getAssocTable">
      <wsdl:input message="impl:getAssocTableRequest" name="getAssocTableRequest"/>
      <wsdl:output message="impl:getAssocTableResponse" name="getAssocTableResponse"/>
      <wsdl:fault message="impl:NoSuchNameExceptionResponse"
                  name="NoSuchNameExceptionFault"/>
      <wsdl:fault message="impl:SecurityExceptionResponse"
                  name="SecurityExceptionFault"/>
      <wsdl:fault message="impl:ImplementationExceptionResponse"
                  name="ImplementationExceptionFault"/>
    </wsdl:operation>

    <wsdl:operation name="putAssocTableEntries">
      <wsdl:input message="impl:putAssocTableEntriesRequest"
                  name="putAssocTableEntriesRequest"/>
      <wsdl:output message="impl:putAssocTableEntriesResponse"
                   name="putAssocTableEntriesResponse"/>
      <wsdl:fault message="impl:NoSuchNameExceptionResponse"
                  name="NoSuchNameExceptionFault"/>
      <wsdl:fault message="impl:InvalidAssocTableEntryExceptionResponse"
                  name="InvalidAssocTableEntryExceptionFault"/>
      <wsdl:fault message="impl:SecurityExceptionResponse"
                  name="SecurityExceptionFault"/>
      <wsdl:fault message="impl:ImplementationExceptionResponse"
                  name="ImplementationExceptionFault"/>
    </wsdl:operation>

    <wsdl:operation name="getAssocTableValue">
      <wsdl:input message="impl:getAssocTableValueRequest"
                  name="getAssocTableValueRequest"/>
      <wsdl:output message="impl:getAssocTableValueResponse"
                   name="getAssocTableValueResponse"/>
      <wsdl:fault message="impl:NoSuchNameExceptionResponse"
                  name="NoSuchNameExceptionFault"/>
      <wsdl:fault message="impl:InvalidEPCExceptionResponse"
                  name="InvalidEPCExceptionFault"/>
      <wsdl:fault message="impl:SecurityExceptionResponse"
                  name="SecurityExceptionFault"/>
      <wsdl:fault message="impl:ImplementationExceptionResponse"
                  name="ImplementationExceptionFault"/>
    </wsdl:operation>

    <wsdl:operation name="getAssocTableEntries">
      <wsdl:input message="impl:getAssocTableEntriesRequest"
                  name="getAssocTableEntriesRequest"/>
      <wsdl:output message="impl:getAssocTableEntriesResponse"
                   name="getAssocTableEntriesResponse"/>
      <wsdl:fault message="impl:NoSuchNameExceptionResponse"
                  name="NoSuchNameExceptionFault"/>
      <wsdl:fault message="impl:InvalidPatternExceptionResponse"
                  name="InvalidPatternExceptionFault"/>
      <wsdl:fault message="impl:SecurityExceptionResponse"
                  name="SecurityExceptionFault"/>
      <wsdl:fault message="impl:ImplementationExceptionResponse"
                  name="ImplementationExceptionFault"/>
    </wsdl:operation>
```

```

      <wsdl:operation name="removeAssocTableEntry">
        <wsdl:input message="impl:removeAssocTableEntryRequest"
                    name="removeAssocTableEntryRequest"/>
        <wsdl:output message="impl:removeAssocTableEntryResponse"
                     name="removeAssocTableEntryResponse"/>
        <wsdl:fault message="impl:NoSuchNameExceptionResponse"
                    name="NoSuchNameExceptionFault"/>
        <wsdl:fault message="impl:InvalidEPCExceptionResponse"
                    name="InvalidEPCExceptionFault"/>
        <wsdl:fault message="impl:SecurityExceptionResponse"
                    name="SecurityExceptionFault"/>
        <wsdl:fault message="impl:ImplementationExceptionResponse"
                    name="ImplementationExceptionFault"/>
      </wsdl:operation>

      <wsdl:operation name="removeAssocTableEntries">
        <wsdl:input message="impl:removeAssocTableEntriesRequest"
                    name="removeAssocTableEntriesRequest"/>
        <wsdl:output message="impl:removeAssocTableEntriesResponse"
                     name="removeAssocTableEntriesResponse"/>
        <wsdl:fault message="impl:NoSuchNameExceptionResponse"
                    name="NoSuchNameExceptionFault"/>
        <wsdl:fault message="impl:InvalidPatternExceptionResponse"
                    name="InvalidPatternExceptionFault"/>
        <wsdl:fault message="impl:SecurityExceptionResponse"
                    name="SecurityExceptionFault"/>
        <wsdl:fault message="impl:ImplementationExceptionResponse"
                    name="ImplementationExceptionFault"/>
      </wsdl:operation>

      <wsdl:operation name="defineRNG">
        <wsdl:input message="impl:defineRNGRequest" name="defineRNGRequest"/>
        <wsdl:output message="impl:defineRNGResponse" name="defineRNGResponse"/>
        <wsdl:fault message="impl:DuplicateNameExceptionResponse"
                    name="DuplicateNameExceptionFault"/>
        <wsdl:fault message="impl:RNGValidationExceptionResponse"
                    name="RNGValidationExceptionFault"/>
        <wsdl:fault message="impl:SecurityExceptionResponse"
                    name="SecurityExceptionFault"/>
        <wsdl:fault message="impl:ImplementationExceptionResponse"
                    name="ImplementationExceptionFault"/>
      </wsdl:operation>

      <wsdl:operation name="undefineRNG">
        <wsdl:input message="impl:undefineRNGRequest" name="undefineRNGRequest"/>
        <wsdl:output message="impl:undefineRNGResponse" name="undefineRNGResponse"/>
        <wsdl:fault message="impl:NoSuchNameExceptionResponse"
                    name="NoSuchNameExceptionFault"/>
        <wsdl:fault message="impl:InUseExceptionResponse" name="InUseExceptionFault"/>
        <wsdl:fault message="impl:SecurityExceptionResponse"
                    name="SecurityExceptionFault"/>
        <wsdl:fault message="impl:ImplementationExceptionResponse"
                    name="ImplementationExceptionFault"/>
      </wsdl:operation>

      <wsdl:operation name="getRNGNames">
        <wsdl:input message="impl:getRNGNamesRequest" name="getRNGNamesRequest"/>
        <wsdl:output message="impl:getRNGNamesResponse" name="getRNGNamesResponse"/>
        <wsdl:fault message="impl:SecurityExceptionResponse"
                    name="SecurityExceptionFault"/>
        <wsdl:fault message="impl:ImplementationExceptionResponse"
                    name="ImplementationExceptionFault"/>
      </wsdl:operation>

      <wsdl:operation name="getRNG">
        <wsdl:input message="impl:getRNGRequest" name="getRNGRequest"/>
        <wsdl:output message="impl:getRNGResponse" name="getRNGResponse"/>
        <wsdl:fault message="impl:NoSuchNameExceptionResponse"
                    name="NoSuchNameExceptionFault"/>
      <wsdl:fault message="impl:SecurityExceptionResponse"
                  name="SecurityExceptionFault"/>
      <wsdl:fault message="impl:ImplementationExceptionResponse"
                  name="ImplementationExceptionFault"/>
    </wsdl:operation>
  </wsdl:portType>
  <!-- ALECCSERVICE BINDING -->

  <wsdl:binding name="ALECCServiceBinding" type="impl:ALECCServicePortType">
    <wsdlsoap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>

    <wsdl:operation name="define">
      <wsdlsoap:operation soapAction=""/>
      <wsdl:input name="defineRequest">
        <wsdlsoap:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="defineResponse">
        <wsdlsoap:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="DuplicateNameExceptionFault">
        <wsdlsoap:fault name="DuplicateNameExceptionFault" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="CCSpecValidationExceptionFault">
        <wsdlsoap:fault name="CCSpecValidationExceptionFault" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="SecurityExceptionFault">
        <wsdlsoap:fault name="SecurityExceptionFault" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="ImplementationExceptionFault">
        <wsdlsoap:fault name="ImplementationExceptionFault" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>

    <wsdl:operation name="undefine">
      <wsdlsoap:operation soapAction=""/>
      <wsdl:input name="undefineRequest">
        <wsdlsoap:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="undefineResponse">
        <wsdlsoap:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="NoSuchNameExceptionFault">
        <wsdlsoap:fault name="NoSuchNameExceptionFault" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="SecurityExceptionFault">
        <wsdlsoap:fault name="SecurityExceptionFault" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="ImplementationExceptionFault">
        <wsdlsoap:fault name="ImplementationExceptionFault" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>

    <wsdl:operation name="getCCSpec">
      <wsdlsoap:operation soapAction=""/>
      <wsdl:input name="getCCSpecRequest">
        <wsdlsoap:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="getCCSpecResponse">
        <wsdlsoap:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="NoSuchNameExceptionFault">
        <wsdlsoap:fault name="NoSuchNameExceptionFault" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="SecurityExceptionFault">
        <wsdlsoap:fault name="SecurityExceptionFault" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="ImplementationExceptionFault">
        <wsdlsoap:fault name="ImplementationExceptionFault" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>
```

```
      <wsdl:operation name="getCCSpecNames">
        <wsdlsoap:operation soapAction=""/>
        <wsdl:input name="getCCSpecNamesRequest">
          <wsdlsoap:body use="literal"/>
        </wsdl:input>
        <wsdl:output name="getCCSpecNamesResponse">
          <wsdlsoap:body use="literal"/>
        </wsdl:output>
        <wsdl:fault name="SecurityExceptionFault">
          <wsdlsoap:fault name="SecurityExceptionFault" use="literal"/>
        </wsdl:fault>
        <wsdl:fault name="ImplementationExceptionFault">
          <wsdlsoap:fault name="ImplementationExceptionFault" use="literal"/>
        </wsdl:fault>
      </wsdl:operation>

      <wsdl:operation name="subscribe">
        <wsdlsoap:operation soapAction=""/>
        <wsdl:input name="subscribeRequest">
          <wsdlsoap:body use="literal"/>
        </wsdl:input>
        <wsdl:output name="subscribeResponse">
          <wsdlsoap:body use="literal"/>
        </wsdl:output>
        <wsdl:fault name="NoSuchNameExceptionFault">
          <wsdlsoap:fault name="NoSuchNameExceptionFault" use="literal"/>
        </wsdl:fault>
        <wsdl:fault name="InvalidURIExceptionFault">
          <wsdlsoap:fault name="InvalidURIExceptionFault" use="literal"/>
        </wsdl:fault>
        <wsdl:fault name="DuplicateSubscriptionExceptionFault">
          <wsdlsoap:fault name="DuplicateSubscriptionExceptionFault" use="literal"/>
        </wsdl:fault>
        <wsdl:fault name="ParameterForbiddenExceptionFault">
          <wsdlsoap:fault name="ParameterForbiddenExceptionFault" use="literal"/>
        </wsdl:fault>
        <wsdl:fault name="SecurityExceptionFault">
          <wsdlsoap:fault name="SecurityExceptionFault" use="literal"/>
        </wsdl:fault>
        <wsdl:fault name="ImplementationExceptionFault">
          <wsdlsoap:fault name="ImplementationExceptionFault" use="literal"/>
        </wsdl:fault>
      </wsdl:operation>

      <wsdl:operation name="unsubscribe">
        <wsdlsoap:operation soapAction=""/>
        <wsdl:input name="unsubscribeRequest">
          <wsdlsoap:body use="literal"/>
        </wsdl:input>
        <wsdl:output name="unsubscribeResponse">
          <wsdlsoap:body use="literal"/>
        </wsdl:output>
        <wsdl:fault name="NoSuchNameExceptionFault">
          <wsdlsoap:fault name="NoSuchNameExceptionFault" use="literal"/>
        </wsdl:fault>
        <wsdl:fault name="NoSuchSubscriberExceptionFault">
          <wsdlsoap:fault name="NoSuchSubscriberExceptionFault" use="literal"/>
        </wsdl:fault>
        <wsdl:fault name="InvalidURIExceptionFault">
          <wsdlsoap:fault name="InvalidURIExceptionFault" use="literal"/>
        </wsdl:fault>
        <wsdl:fault name="SecurityExceptionFault">
          <wsdlsoap:fault name="SecurityExceptionFault" use="literal"/>
        </wsdl:fault>
        <wsdl:fault name="ImplementationExceptionFault">
          <wsdlsoap:fault name="ImplementationExceptionFault" use="literal"/>
        </wsdl:fault>
      </wsdl:operation>
```

```
    <wsdl:operation name="poll">
      <wsdlsoap:operation soapAction=""/>
      <wsdl:input name="pollRequest">
        <wsdlsoap:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="pollResponse">
        <wsdlsoap:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="NoSuchNameExceptionFault">
        <wsdlsoap:fault name="NoSuchNameExceptionFault" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="ParameterExceptionFault">
        <wsdlsoap:fault name="ParameterExceptionFault" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="SecurityExceptionFault">
        <wsdlsoap:fault name="SecurityExceptionFault" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="ImplementationExceptionFault">
        <wsdlsoap:fault name="ImplementationExceptionFault" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>

    <wsdl:operation name="immediate">
      <wsdlsoap:operation soapAction=""/>
      <wsdl:input name="immediateRequest">
        <wsdlsoap:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="immediateResponse">
        <wsdlsoap:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="CCSpecValidationExceptionFault">
        <wsdlsoap:fault name="CCSpecValidationExceptionFault" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="ParameterForbiddenExceptionFault">
        <wsdlsoap:fault name="ParameterForbiddenExceptionFault" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="SecurityExceptionFault">
        <wsdlsoap:fault name="SecurityExceptionFault" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="ImplementationExceptionFault">
        <wsdlsoap:fault name="ImplementationExceptionFault" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>

    <wsdl:operation name="getSubscribers">
      <wsdlsoap:operation soapAction=""/>
      <wsdl:input name="getSubscribersRequest">
        <wsdlsoap:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="getSubscribersResponse">
        <wsdlsoap:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="NoSuchNameExceptionFault">
        <wsdlsoap:fault name="NoSuchNameExceptionFault" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="SecurityExceptionFault">
        <wsdlsoap:fault name="SecurityExceptionFault" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="ImplementationExceptionFault">
        <wsdlsoap:fault name="ImplementationExceptionFault" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>

    <wsdl:operation name="getStandardVersion">
      <wsdlsoap:operation soapAction=""/>
      <wsdl:input name="getStandardVersionRequest">
        <wsdlsoap:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="getStandardVersionResponse">
        <wsdlsoap:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="ImplementationExceptionFault">
        <wsdlsoap:fault name="ImplementationExceptionFault" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>

    <wsdl:operation name="getVendorVersion">
      <wsdlsoap:operation soapAction=""/>
      <wsdl:input name="getVendorVersionRequest">
        <wsdlsoap:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="getVendorVersionResponse">
        <wsdlsoap:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="ImplementationExceptionFault">
        <wsdlsoap:fault name="ImplementationExceptionFault" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>

    <wsdl:operation name="defineEPCCache">
      <wsdlsoap:operation soapAction=""/>
      <wsdl:input name="defineEPCCacheRequest">
        <wsdlsoap:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="defineEPCCacheResponse">
        <wsdlsoap:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="DuplicateNameExceptionFault">
        <wsdlsoap:fault name="DuplicateNameExceptionFault" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="EPCCacheSpecValidationExceptionFault">
        <wsdlsoap:fault name="EPCCacheSpecValidationExceptionFault" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="InvalidPatternExceptionFault">
        <wsdlsoap:fault name="InvalidPatternExceptionFault" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="SecurityExceptionFault">
        <wsdlsoap:fault name="SecurityExceptionFault" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="ImplementationExceptionFault">
        <wsdlsoap:fault name="ImplementationExceptionFault" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>

    <wsdl:operation name="undefineEPCCache">
      <wsdlsoap:operation soapAction=""/>
      <wsdl:input name="undefineEPCCacheRequest">
        <wsdlsoap:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="undefineEPCCacheResponse">
        <wsdlsoap:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="NoSuchNameExceptionFault">
        <wsdlsoap:fault name="NoSuchNameExceptionFault" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="InUseExceptionFault">
        <wsdlsoap:fault name="InUseExceptionFault" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="SecurityExceptionFault">
        <wsdlsoap:fault name="SecurityExceptionFault" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="ImplementationExceptionFault">
        <wsdlsoap:fault name="ImplementationExceptionFault" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>

    <wsdl:operation name="getEPCCache">
      <wsdlsoap:operation soapAction=""/>
      <wsdl:input name="getEPCCacheRequest">
        <wsdlsoap:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="getEPCCacheResponse">
        <wsdlsoap:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="NoSuchNameExceptionFault">
        <wsdlsoap:fault name="NoSuchNameExceptionFault" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="SecurityExceptionFault">
        <wsdlsoap:fault name="SecurityExceptionFault" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="ImplementationExceptionFault">
        <wsdlsoap:fault name="ImplementationExceptionFault" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>

    <wsdl:operation name="getEPCCacheNames">
      <wsdlsoap:operation soapAction=""/>
      <wsdl:input name="getEPCCacheNamesRequest">
        <wsdlsoap:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="getEPCCacheNamesResponse">
        <wsdlsoap:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="SecurityExceptionFault">
        <wsdlsoap:fault name="SecurityExceptionFault" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="ImplementationExceptionFault">
        <wsdlsoap:fault name="ImplementationExceptionFault" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>

    <wsdl:operation name="replenishEPCCache">
      <wsdlsoap:operation soapAction=""/>
      <wsdl:input name="replenishEPCCacheRequest">
        <wsdlsoap:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="replenishEPCCacheResponse">
        <wsdlsoap:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="NoSuchNameExceptionFault">
        <wsdlsoap:fault name="NoSuchNameExceptionFault" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="InvalidPatternExceptionFault">
        <wsdlsoap:fault name="InvalidPatternExceptionFault" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="SecurityExceptionFault">
        <wsdlsoap:fault name="SecurityExceptionFault" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="ImplementationExceptionFault">
        <wsdlsoap:fault name="ImplementationExceptionFault" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>

    <wsdl:operation name="depleteEPCCache">
      <wsdlsoap:operation soapAction=""/>
      <wsdl:input name="depleteEPCCacheRequest">
        <wsdlsoap:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="depleteEPCCacheResponse">
        <wsdlsoap:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="NoSuchNameExceptionFault">
        <wsdlsoap:fault name="NoSuchNameExceptionFault" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="SecurityExceptionFault">
        <wsdlsoap:fault name="SecurityExceptionFault" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="ImplementationExceptionFault">
        <wsdlsoap:fault name="ImplementationExceptionFault" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>

    <wsdl:operation name="getEPCCacheContents">
      <wsdlsoap:operation soapAction=""/>
      <wsdl:input name="getEPCCacheContentsRequest">
        <wsdlsoap:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="getEPCCacheContentsResponse">
        <wsdlsoap:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="NoSuchNameExceptionFault">
        <wsdlsoap:fault name="NoSuchNameExceptionFault" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="SecurityExceptionFault">
        <wsdlsoap:fault name="SecurityExceptionFault" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="ImplementationExceptionFault">
        <wsdlsoap:fault name="ImplementationExceptionFault" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>

    <wsdl:operation name="defineAssocTable">
      <wsdlsoap:operation soapAction=""/>
      <wsdl:input name="defineAssocTableRequest">
        <wsdlsoap:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="defineAssocTableResponse">
        <wsdlsoap:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="DuplicateNameExceptionFault">
        <wsdlsoap:fault name="DuplicateNameExceptionFault" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="AssocTableValidationExceptionFault">
        <wsdlsoap:fault name="AssocTableValidationExceptionFault" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="InvalidAssocTableEntryExceptionFault">
        <wsdlsoap:fault name="InvalidAssocTableEntryExceptionFault" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="SecurityExceptionFault">
        <wsdlsoap:fault name="SecurityExceptionFault" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="ImplementationExceptionFault">
        <wsdlsoap:fault name="ImplementationExceptionFault" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>

    <wsdl:operation name="undefineAssocTable">
      <wsdlsoap:operation soapAction=""/>
      <wsdl:input name="undefineAssocTableRequest">
        <wsdlsoap:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="undefineAssocTableResponse">
        <wsdlsoap:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="NoSuchNameExceptionFault">
        <wsdlsoap:fault name="NoSuchNameExceptionFault" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="InUseExceptionFault">
        <wsdlsoap:fault name="InUseExceptionFault" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="SecurityExceptionFault">
        <wsdlsoap:fault name="SecurityExceptionFault" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="ImplementationExceptionFault">
        <wsdlsoap:fault name="ImplementationExceptionFault" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>

    <wsdl:operation name="getAssocTableNames">
      <wsdlsoap:operation soapAction=""/>
```

```
        <wsdl:input name="getAssocTableNamesRequest">
          <wsdlsoap:body use="literal"/>
        </wsdl:input>
        <wsdl:output name="getAssocTableNamesResponse">
          <wsdlsoap:body use="literal"/>
        </wsdl:output>
        <wsdl:fault name="SecurityExceptionFault">
          <wsdlsoap:fault name="SecurityExceptionFault" use="literal"/>
        </wsdl:fault>
        <wsdl:fault name="ImplementationExceptionFault">
          <wsdlsoap:fault name="ImplementationExceptionFault" use="literal"/>
        </wsdl:fault>
      </wsdl:operation>

      <wsdl:operation name="getAssocTable">
        <wsdlsoap:operation soapAction=""/>
        <wsdl:input name="getAssocTableRequest">
          <wsdlsoap:body use="literal"/>
        </wsdl:input>
        <wsdl:output name="getAssocTableResponse">
          <wsdlsoap:body use="literal"/>
        </wsdl:output>
        <wsdl:fault name="NoSuchNameExceptionFault">
          <wsdlsoap:fault name="NoSuchNameExceptionFault" use="literal"/>
        </wsdl:fault>
        <wsdl:fault name="SecurityExceptionFault">
          <wsdlsoap:fault name="SecurityExceptionFault" use="literal"/>
        </wsdl:fault>
        <wsdl:fault name="ImplementationExceptionFault">
          <wsdlsoap:fault name="ImplementationExceptionFault" use="literal"/>
        </wsdl:fault>
      </wsdl:operation>

      <wsdl:operation name="putAssocTableEntries">
        <wsdlsoap:operation soapAction=""/>
        <wsdl:input name="putAssocTableEntriesRequest">
          <wsdlsoap:body use="literal"/>
        </wsdl:input>
        <wsdl:output name="putAssocTableEntriesResponse">
          <wsdlsoap:body use="literal"/>
        </wsdl:output>
        <wsdl:fault name="NoSuchNameExceptionFault">
          <wsdlsoap:fault name="NoSuchNameExceptionFault" use="literal"/>
        </wsdl:fault>
        <wsdl:fault name="InvalidAssocTableEntryExceptionFault">
          <wsdlsoap:fault name="InvalidAssocTableEntryExceptionFault" use="literal"/>
        </wsdl:fault>
        <wsdl:fault name="SecurityExceptionFault">
          <wsdlsoap:fault name="SecurityExceptionFault" use="literal"/>
        </wsdl:fault>
        <wsdl:fault name="ImplementationExceptionFault">
          <wsdlsoap:fault name="ImplementationExceptionFault" use="literal"/>
        </wsdl:fault>
      </wsdl:operation>

      <wsdl:operation name="getAssocTableValue">
        <wsdlsoap:operation soapAction=""/>
        <wsdl:input name="getAssocTableValueRequest">
          <wsdlsoap:body use="literal"/>
        </wsdl:input>
        <wsdl:output name="getAssocTableValueResponse">
          <wsdlsoap:body use="literal"/>
        </wsdl:output>
        <wsdl:fault name="NoSuchNameExceptionFault">
          <wsdlsoap:fault name="NoSuchNameExceptionFault" use="literal"/>
        </wsdl:fault>
        <wsdl:fault name="InvalidEPCExceptionFault">
          <wsdlsoap:fault name="InvalidEPCExceptionFault" use="literal"/>
        </wsdl:fault>
        <wsdl:fault name="SecurityExceptionFault">
```

```
          <wsdlsoap:fault name="SecurityExceptionFault" use="literal"/>
        </wsdl:fault>
        <wsdl:fault name="ImplementationExceptionFault">
          <wsdlsoap:fault name="ImplementationExceptionFault" use="literal"/>
        </wsdl:fault>
      </wsdl:operation>

      <wsdl:operation name="getAssocTableEntries">
        <wsdlsoap:operation soapAction=""/>
        <wsdl:input name="getAssocTableEntriesRequest">
          <wsdlsoap:body use="literal"/>
        </wsdl:input>
        <wsdl:output name="getAssocTableEntriesResponse">
          <wsdlsoap:body use="literal"/>
        </wsdl:output>
        <wsdl:fault name="NoSuchNameExceptionFault">
          <wsdlsoap:fault name="NoSuchNameExceptionFault" use="literal"/>
        </wsdl:fault>
        <wsdl:fault name="InvalidPatternExceptionFault">
          <wsdlsoap:fault name="InvalidPatternExceptionFault" use="literal"/>
        </wsdl:fault>
        <wsdl:fault name="SecurityExceptionFault">
          <wsdlsoap:fault name="SecurityExceptionFault" use="literal"/>
        </wsdl:fault>
        <wsdl:fault name="ImplementationExceptionFault">
          <wsdlsoap:fault name="ImplementationExceptionFault" use="literal"/>
        </wsdl:fault>
      </wsdl:operation>

      <wsdl:operation name="removeAssocTableEntry">
        <wsdlsoap:operation soapAction=""/>
        <wsdl:input name="removeAssocTableEntryRequest">
          <wsdlsoap:body use="literal"/>
        </wsdl:input>
        <wsdl:output name="removeAssocTableEntryResponse">
          <wsdlsoap:body use="literal"/>
        </wsdl:output>
        <wsdl:fault name="NoSuchNameExceptionFault">
          <wsdlsoap:fault name="NoSuchNameExceptionFault" use="literal"/>
        </wsdl:fault>
        <wsdl:fault name="InvalidEPCExceptionFault">
          <wsdlsoap:fault name="InvalidEPCExceptionFault" use="literal"/>
        </wsdl:fault>
        <wsdl:fault name="SecurityExceptionFault">
          <wsdlsoap:fault name="SecurityExceptionFault" use="literal"/>
        </wsdl:fault>
        <wsdl:fault name="ImplementationExceptionFault">
          <wsdlsoap:fault name="ImplementationExceptionFault" use="literal"/>
        </wsdl:fault>
      </wsdl:operation>

      <wsdl:operation name="removeAssocTableEntries">
        <wsdlsoap:operation soapAction=""/>
        <wsdl:input name="removeAssocTableEntriesRequest">
          <wsdlsoap:body use="literal"/>
        </wsdl:input>
        <wsdl:output name="removeAssocTableEntriesResponse">
          <wsdlsoap:body use="literal"/>
        </wsdl:output>
        <wsdl:fault name="NoSuchNameExceptionFault">
          <wsdlsoap:fault name="NoSuchNameExceptionFault" use="literal"/>
        </wsdl:fault>
        <wsdl:fault name="InvalidPatternExceptionFault">
          <wsdlsoap:fault name="InvalidPatternExceptionFault" use="literal"/>
        </wsdl:fault>
        <wsdl:fault name="SecurityExceptionFault">
          <wsdlsoap:fault name="SecurityExceptionFault" use="literal"/>
        </wsdl:fault>
        <wsdl:fault name="ImplementationExceptionFault">
          <wsdlsoap:fault name="ImplementationExceptionFault" use="literal"/>
        </wsdl:fault>
      </wsdl:operation>

      <wsdl:operation name="defineRNG">
        <wsdlsoap:operation soapAction=""/>
        <wsdl:input name="defineRNGRequest">
          <wsdlsoap:body use="literal"/>
        </wsdl:input>
        <wsdl:output name="defineRNGResponse">
          <wsdlsoap:body use="literal"/>
        </wsdl:output>
        <wsdl:fault name="DuplicateNameExceptionFault">
          <wsdlsoap:fault name="DuplicateNameExceptionFault" use="literal"/>
        </wsdl:fault>
        <wsdl:fault name="RNGValidationExceptionFault">
          <wsdlsoap:fault name="RNGValidationExceptionFault" use="literal"/>
        </wsdl:fault>
        <wsdl:fault name="SecurityExceptionFault">
          <wsdlsoap:fault name="SecurityExceptionFault" use="literal"/>
        </wsdl:fault>
        <wsdl:fault name="ImplementationExceptionFault">
          <wsdlsoap:fault name="ImplementationExceptionFault" use="literal"/>
        </wsdl:fault>
      </wsdl:operation>

      <wsdl:operation name="undefineRNG">
        <wsdlsoap:operation soapAction=""/>
        <wsdl:input name="undefineRNGRequest">
          <wsdlsoap:body use="literal"/>
        </wsdl:input>
        <wsdl:output name="undefineRNGResponse">
          <wsdlsoap:body use="literal"/>
        </wsdl:output>
        <wsdl:fault name="NoSuchNameExceptionFault">
          <wsdlsoap:fault name="NoSuchNameExceptionFault" use="literal"/>
        </wsdl:fault>
        <wsdl:fault name="InUseExceptionFault">
          <wsdlsoap:fault name="InUseExceptionFault" use="literal"/>
        </wsdl:fault>
        <wsdl:fault name="SecurityExceptionFault">
          <wsdlsoap:fault name="SecurityExceptionFault" use="literal"/>
        </wsdl:fault>
        <wsdl:fault name="ImplementationExceptionFault">
          <wsdlsoap:fault name="ImplementationExceptionFault" use="literal"/>
        </wsdl:fault>
      </wsdl:operation>

      <wsdl:operation name="getRNGNames">
        <wsdlsoap:operation soapAction=""/>
        <wsdl:input name="getRNGNamesRequest">
          <wsdlsoap:body use="literal"/>
        </wsdl:input>
        <wsdl:output name="getRNGNamesResponse">
          <wsdlsoap:body use="literal"/>
        </wsdl:output>
        <wsdl:fault name="SecurityExceptionFault">
          <wsdlsoap:fault name="SecurityExceptionFault" use="literal"/>
        </wsdl:fault>
        <wsdl:fault name="ImplementationExceptionFault">
          <wsdlsoap:fault name="ImplementationExceptionFault" use="literal"/>
        </wsdl:fault>
      </wsdl:operation>

      <wsdl:operation name="getRNG">
        <wsdlsoap:operation soapAction=""/>
        <wsdl:input name="getRNGRequest">
          <wsdlsoap:body use="literal"/>
        </wsdl:input>
        <wsdl:output name="getRNGResponse">
          <wsdlsoap:body use="literal"/>
        </wsdl:output>
        <wsdl:fault name="NoSuchNameExceptionFault">
          <wsdlsoap:fault name="NoSuchNameExceptionFault" use="literal"/>
        </wsdl:fault>
        <wsdl:fault name="SecurityExceptionFault">
          <wsdlsoap:fault name="SecurityExceptionFault" use="literal"/>
        </wsdl:fault>
        <wsdl:fault name="ImplementationExceptionFault">
          <wsdlsoap:fault name="ImplementationExceptionFault" use="literal"/>
        </wsdl:fault>
      </wsdl:operation>
    </wsdl:binding>

    <!-- ALECCSERVICE -->
    <wsdl:service name="ALECCService">
      <wsdl:port binding="impl:ALECCServiceBinding" name="ALECCServicePort">
        <!-- The value of the location attribute below is an example only;
             Implementations are free to choose any appropriate URL. -->
        <wsdlsoap:address location="http://localhost:8080/services/ALECCService"/>
      </wsdl:port>
    </wsdl:service>
  </wsdl:definitions>
```

4.5 ALE Tag Memory API SOAP Binding

The following is a Web Services Description Language (WSDL) 1.1 [WSDL1.1] specification defining the standard SOAP 1.1 [SOAP1.1] binding of the ALE Tag Memory API. This SOAP binding is compliant with the WS-I Basic Profile Version 1.0 [WSI].

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- ALETMSERVICE DEFINITIONS -->
<wsdl:definitions xmlns:xsd="http://www.w3.org/2001/XMLSchema"
                  xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
                  xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/"
                  xmlns:ale="urn:epcglobal:ale:xsd:1"
                  xmlns:impl="urn:epcglobal:aletm:wsdl:1"
                  targetNamespace="urn:epcglobal:aletm:wsdl:1">
  <!-- ALETMSERVICE TYPES -->
  <wsdl:types>
    <xsd:schema targetNamespace="urn:epcglobal:aletm:wsdl:1">
      <xsd:import namespace="urn:epcglobal:ale:xsd:1"
                  schemaLocation="EPCglobal-ale-1_1-aletm.xsd"/>
      <!-- ALETMSERVICE MESSAGE WRAPPERS -->

      <xsd:element name="DefineTMSpec" type="impl:DefineTMSpec"/>
      <xsd:complexType name="DefineTMSpec">
        <xsd:sequence>
          <xsd:element name="specName" type="xsd:string"/>
          <xsd:element name="spec" type="ale:TMSpec"/>
        </xsd:sequence>
      </xsd:complexType>
      <xsd:element name="DefineTMSpecResult">
        <xsd:complexType/>
      </xsd:element>

      <xsd:element name="UndefineTMSpec" type="impl:UndefineTMSpec"/>
      <xsd:complexType name="UndefineTMSpec">
        <xsd:sequence>
          <xsd:element name="specName" type="xsd:string"/>
        </xsd:sequence>
      </xsd:complexType>
      <xsd:element name="UndefineTMSpecResult">
        <xsd:complexType/>
      </xsd:element>

      <xsd:element name="GetTMSpec" type="impl:GetTMSpec"/>
      <xsd:complexType name="GetTMSpec">
        <xsd:sequence>
          <xsd:element name="specName" type="xsd:string"/>
        </xsd:sequence>
      </xsd:complexType>
      <xsd:element name="GetTMSpecResult" type="ale:TMSpec"/>

      <xsd:element name="GetTMSpecNames" type="impl:EmptyParms"/>
      <xsd:element name="GetTMSpecNamesResult" type="impl:ArrayOfString"/>

      <xsd:element name="GetStandardVersion" type="impl:EmptyParms"/>
      <xsd:element name="GetStandardVersionResult" type="xsd:string"/>

      <xsd:element name="GetVendorVersion" type="impl:EmptyParms"/>
      <xsd:element name="GetVendorVersionResult" type="xsd:string"/>

      <xsd:element name="ALEException" type="impl:ALEException"/>
      <xsd:complexType name="ALEException">
        <xsd:sequence>
          <xsd:element name="reason" type="xsd:string"/>
        </xsd:sequence>
      </xsd:complexType>

      <xsd:element name="SecurityException" type="impl:SecurityException"/>
      <xsd:complexType name="SecurityException">
        <xsd:complexContent>
          <xsd:extension base="impl:ALEException"/>
        </xsd:complexContent>
      </xsd:complexType>

      <xsd:element name="DuplicateNameException" type="impl:DuplicateNameException"/>
      <xsd:complexType name="DuplicateNameException">
        <xsd:complexContent>
          <xsd:extension base="impl:ALEException"/>
        </xsd:complexContent>
      </xsd:complexType>

      <xsd:element name="TMSpecValidationException"
                   type="impl:TMSpecValidationException"/>
      <xsd:complexType name="TMSpecValidationException">
        <xsd:complexContent>
          <xsd:extension base="impl:ALEException"/>
        </xsd:complexContent>
      </xsd:complexType>

      <xsd:element name="NoSuchNameException" type="impl:NoSuchNameException"/>
      <xsd:complexType name="NoSuchNameException">
        <xsd:complexContent>
          <xsd:extension base="impl:ALEException"/>
        </xsd:complexContent>
      </xsd:complexType>

      <xsd:element name="InUseException" type="impl:InUseException"/>
      <xsd:complexType name="InUseException">
        <xsd:complexContent>
          <xsd:extension base="impl:ALEException"/>
        </xsd:complexContent>
      </xsd:complexType>

      <xsd:element name="ImplementationException" type="impl:ImplementationException"/>
      <xsd:complexType name="ImplementationException">
        <xsd:complexContent>
          <xsd:extension base="impl:ALEException">
            <xsd:sequence>
              <xsd:element name="severity" type="impl:ImplementationExceptionSeverity"/>
            </xsd:sequence>
          </xsd:extension>
        </xsd:complexContent>
      </xsd:complexType>

      <xsd:complexType name="ArrayOfString">
```

```
          <xsd:sequence>
            <xsd:element name="string" type="xsd:string" minOccurs="0"
                         maxOccurs="unbounded"/>
          </xsd:sequence>
        </xsd:complexType>

        <!-- The ImplementationExceptionSeverity type is an enumerated type.
             The following strings are legal values for this type:
               ERROR
               SEVERE
              -->
        <xsd:simpleType name="ImplementationExceptionSeverity">
          <xsd:restriction base="xsd:string"/>
        </xsd:simpleType>

        <xsd:complexType name="EmptyParms"/>
      </xsd:schema>
    </wsdl:types>
    <!-- ALETMSERVICE MESSAGES -->

    <wsdl:message name="defineTMSpecRequest">
      <wsdl:part name="parms" element="impl:DefineTMSpec"/>
    </wsdl:message>
    <wsdl:message name="defineTMSpecResponse">
      <wsdl:part name="defineTMSpecReturn" element="impl:DefineTMSpecResult"/>
    </wsdl:message>

    <wsdl:message name="undefineTMSpecRequest">
      <wsdl:part name="parms" element="impl:UndefineTMSpec"/>
    </wsdl:message>
    <wsdl:message name="undefineTMSpecResponse">
      <wsdl:part name="undefineTMSpecReturn" element="impl:UndefineTMSpecResult"/>
    </wsdl:message>

    <wsdl:message name="getTMSpecRequest">
      <wsdl:part name="parms" element="impl:GetTMSpec"/>
    </wsdl:message>
    <wsdl:message name="getTMSpecResponse">
      <wsdl:part name="getTMSpecReturn" element="impl:GetTMSpecResult"/>
    </wsdl:message>

    <wsdl:message name="getTMSpecNamesRequest">
      <wsdl:part name="parms" element="impl:GetTMSpecNames"/>
    </wsdl:message>
    <wsdl:message name="getTMSpecNamesResponse">
      <wsdl:part name="getTMSpecNamesReturn" element="impl:GetTMSpecNamesResult"/>
    </wsdl:message>

    <wsdl:message name="getStandardVersionRequest">
      <wsdl:part name="parms" element="impl:GetStandardVersion"/>
    </wsdl:message>
    <wsdl:message name="getStandardVersionResponse">
      <wsdl:part name="getStandardVersionReturn" element="impl:GetStandardVersionResult"/>
    </wsdl:message>

    <wsdl:message name="getVendorVersionRequest">
      <wsdl:part name="parms" element="impl:GetVendorVersion"/>
    </wsdl:message>
    <wsdl:message name="getVendorVersionResponse">
      <wsdl:part name="getVendorVersionReturn" element="impl:GetVendorVersionResult"/>
    </wsdl:message>

    <wsdl:message name="SecurityExceptionResponse">
      <wsdl:part name="fault" element="impl:SecurityException"/>
    </wsdl:message>

    <wsdl:message name="DuplicateNameExceptionResponse">
      <wsdl:part name="fault" element="impl:DuplicateNameException"/>
    </wsdl:message>

```

```
<wsdl:message name="InUseExceptionResponse">
  <wsdl:part name="fault" element="impl:InUseException"/>
</wsdl:message>

<wsdl:message name="TMSpecValidationExceptionResponse">
  <wsdl:part name="fault" element="impl:TMSpecValidationException"/>
</wsdl:message>

<wsdl:message name="NoSuchNameExceptionResponse">
  <wsdl:part name="fault" element="impl:NoSuchNameException"/>
</wsdl:message>

<wsdl:message name="ImplementationExceptionResponse">
  <wsdl:part name="fault" element="impl:ImplementationException"/>
</wsdl:message>
<!-- ALETMSERVICE PORTTYPE -->

<wsdl:portType name="ALETMServicePortType">

  <wsdl:operation name="defineTMSpec">
    <wsdl:input message="impl:defineTMSpecRequest" name="defineTMSpecRequest"/>
    <wsdl:output message="impl:defineTMSpecResponse" name="defineTMSpecResponse"/>
    <wsdl:fault message="impl:DuplicateNameExceptionResponse"
                name="DuplicateNameExceptionFault"/>
    <wsdl:fault message="impl:TMSpecValidationExceptionResponse"
                name="TMSpecValidationExceptionFault"/>
    <wsdl:fault message="impl:SecurityExceptionResponse"
                name="SecurityExceptionFault"/>
    <wsdl:fault message="impl:ImplementationExceptionResponse"
                name="ImplementationExceptionFault"/>
  </wsdl:operation>

  <wsdl:operation name="undefineTMSpec">
    <wsdl:input message="impl:undefineTMSpecRequest" name="undefineTMSpecRequest"/>
    <wsdl:output message="impl:undefineTMSpecResponse" name="undefineTMSpecResponse"/>
    <wsdl:fault message="impl:NoSuchNameExceptionResponse"
                name="NoSuchNameExceptionFault"/>
    <wsdl:fault message="impl:InUseExceptionResponse" name="InUseExceptionFault"/>
    <wsdl:fault message="impl:SecurityExceptionResponse"
                name="SecurityExceptionFault"/>
    <wsdl:fault message="impl:ImplementationExceptionResponse"
                name="ImplementationExceptionFault"/>
  </wsdl:operation>

  <wsdl:operation name="getTMSpec">
    <wsdl:input message="impl:getTMSpecRequest" name="getTMSpecRequest"/>
    <wsdl:output message="impl:getTMSpecResponse" name="getTMSpecResponse"/>
    <wsdl:fault message="impl:NoSuchNameExceptionResponse"
                name="NoSuchNameExceptionFault"/>
    <wsdl:fault message="impl:SecurityExceptionResponse"
                name="SecurityExceptionFault"/>
    <wsdl:fault message="impl:ImplementationExceptionResponse"
                name="ImplementationExceptionFault"/>
  </wsdl:operation>

  <wsdl:operation name="getTMSpecNames">
    <wsdl:input message="impl:getTMSpecNamesRequest" name="getTMSpecNamesRequest"/>
    <wsdl:output message="impl:getTMSpecNamesResponse" name="getTMSpecNamesResponse"/>
    <wsdl:fault message="impl:SecurityExceptionResponse"
                name="SecurityExceptionFault"/>
    <wsdl:fault message="impl:ImplementationExceptionResponse"
                name="ImplementationExceptionFault"/>
  </wsdl:operation>

  <wsdl:operation name="getStandardVersion">
    <wsdl:input message="impl:getStandardVersionRequest"
                name="getStandardVersionRequest"/>
    <wsdl:output message="impl:getStandardVersionResponse"
                 name="getStandardVersionResponse"/>
    <wsdl:fault message="impl:ImplementationExceptionResponse"
```

```
                 name="ImplementationExceptionFault"/>
    </wsdl:operation>

    <wsdl:operation name="getVendorVersion">
      <wsdl:input message="impl:getVendorVersionRequest" name="getVendorVersionRequest"/>
      <wsdl:output message="impl:getVendorVersionResponse"
                  name="getVendorVersionResponse"/>
      <wsdl:fault message="impl:ImplementationExceptionResponse"
                 name="ImplementationExceptionFault"/>
    </wsdl:operation>
  </wsdl:portType>
  <!-- ALETMSERVICE BINDING -->

  <wsdl:binding name="ALETMServiceBinding" type="impl:ALETMServicePortType">
    <wsdlsoap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>

    <wsdl:operation name="defineTMSpec">
      <wsdlsoap:operation soapAction=""/>
      <wsdl:input name="defineTMSpecRequest">
        <wsdlsoap:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="defineTMSpecResponse">
        <wsdlsoap:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="DuplicateNameExceptionFault">
        <wsdlsoap:fault name="DuplicateNameExceptionFault" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="TMSpecValidationExceptionFault">
        <wsdlsoap:fault name="TMSpecValidationExceptionFault" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="SecurityExceptionFault">
        <wsdlsoap:fault name="SecurityExceptionFault" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="ImplementationExceptionFault">
        <wsdlsoap:fault name="ImplementationExceptionFault" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>

    <wsdl:operation name="undefineTMSpec">
      <wsdlsoap:operation soapAction=""/>
      <wsdl:input name="undefineTMSpecRequest">
        <wsdlsoap:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="undefineTMSpecResponse">
        <wsdlsoap:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="NoSuchNameExceptionFault">
        <wsdlsoap:fault name="NoSuchNameExceptionFault" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="InUseExceptionFault">
        <wsdlsoap:fault name="InUseExceptionFault" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="SecurityExceptionFault">
        <wsdlsoap:fault name="SecurityExceptionFault" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="ImplementationExceptionFault">
        <wsdlsoap:fault name="ImplementationExceptionFault" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>

    <wsdl:operation name="getTMSpec">
      <wsdlsoap:operation soapAction=""/>
      <wsdl:input name="getTMSpecRequest">
        <wsdlsoap:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="getTMSpecResponse">
        <wsdlsoap:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="NoSuchNameExceptionFault">
        <wsdlsoap:fault name="NoSuchNameExceptionFault" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="SecurityExceptionFault">
        <wsdlsoap:fault name="SecurityExceptionFault" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="ImplementationExceptionFault">
        <wsdlsoap:fault name="ImplementationExceptionFault" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>

    <wsdl:operation name="getTMSpecNames">
      <wsdlsoap:operation soapAction=""/>
      <wsdl:input name="getTMSpecNamesRequest">
        <wsdlsoap:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="getTMSpecNamesResponse">
        <wsdlsoap:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="SecurityExceptionFault">
        <wsdlsoap:fault name="SecurityExceptionFault" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="ImplementationExceptionFault">
        <wsdlsoap:fault name="ImplementationExceptionFault" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>

    <wsdl:operation name="getStandardVersion">
      <wsdlsoap:operation soapAction=""/>
      <wsdl:input name="getStandardVersionRequest">
        <wsdlsoap:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="getStandardVersionResponse">
        <wsdlsoap:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="ImplementationExceptionFault">
        <wsdlsoap:fault name="ImplementationExceptionFault" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>

    <wsdl:operation name="getVendorVersion">
      <wsdlsoap:operation soapAction=""/>
      <wsdl:input name="getVendorVersionRequest">
        <wsdlsoap:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="getVendorVersionResponse">
        <wsdlsoap:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="ImplementationExceptionFault">
        <wsdlsoap:fault name="ImplementationExceptionFault" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>
  </wsdl:binding>

  <!-- ALETMSERVICE -->
  <wsdl:service name="ALETMService">
    <wsdl:port binding="impl:ALETMServiceBinding" name="ALETMServicePort">
      <!-- The value of the location attribute below is an example only;
           Implementations are free to choose any appropriate URL. -->
      <wsdlsoap:address location="http://localhost:8080/services/ALETMService"/>
    </wsdl:port>
  </wsdl:service>
</wsdl:definitions>
```

4.6 ALE Logical Reader API SOAP Binding

The following is a Web Services Description Language (WSDL) 1.1 [WSDL1.1] specification defining the standard SOAP 1.1 [SOAP1.1] binding of the ALE Logical

Reader API. This SOAP binding is compliant with the WS-I Basic Profile Version 1.0 [WSI].

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- ALELRSERVICE DEFINITIONS -->
<wsdl:definitions xmlns:xsd="http://www.w3.org/2001/XMLSchema"
                  xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
                  xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/"
                  xmlns:ale="urn:epcglobal:ale:xsd:1"
                  xmlns:impl="urn:epcglobal:alelr:wsdl:1"
                  targetNamespace="urn:epcglobal:alelr:wsdl:1">
  <!-- ALELRSERVICE TYPES -->
  <wsdl:types>
    <xsd:schema targetNamespace="urn:epcglobal:alelr:wsdl:1">
      <xsd:import namespace="urn:epcglobal:ale:xsd:1"
                  schemaLocation="EPCglobal-ale-1_1-alelr.xsd"/>
      <!-- ALELRSERVICE MESSAGE WRAPPERS -->

      <xsd:element name="Define" type="impl:Define"/>
      <xsd:complexType name="Define">
        <xsd:sequence>
          <xsd:element name="name" type="xsd:string"/>
          <xsd:element name="spec" type="ale:LRSpec"/>
        </xsd:sequence>
      </xsd:complexType>
      <xsd:element name="DefineResult">
        <xsd:complexType/>
      </xsd:element>

      <xsd:element name="Update" type="impl:Update"/>
      <xsd:complexType name="Update">
        <xsd:sequence>
          <xsd:element name="name" type="xsd:string"/>
          <xsd:element name="spec" type="ale:LRSpec"/>
        </xsd:sequence>
      </xsd:complexType>
      <xsd:element name="UpdateResult">
        <xsd:complexType/>
      </xsd:element>

      <xsd:element name="Undefine" type="impl:Undefine"/>
      <xsd:complexType name="Undefine">
        <xsd:sequence>
          <xsd:element name="name" type="xsd:string"/>
        </xsd:sequence>
      </xsd:complexType>
      <xsd:element name="UndefineResult">
        <xsd:complexType/>
      </xsd:element>

      <xsd:element name="GetLogicalReaderNames" type="impl:EmptyParms"/>
      <xsd:element name="GetLogicalReaderNamesResult" type="impl:ArrayOfString"/>

      <xsd:element name="GetLRSpec" type="impl:GetLRSpec"/>
      <xsd:complexType name="GetLRSpec">
        <xsd:sequence>
          <xsd:element name="name" type="xsd:string"/>
        </xsd:sequence>
      </xsd:complexType>
      <xsd:element name="GetLRSpecResult" type="ale:LRSpec"/>

      <xsd:element name="AddReaders" type="impl:AddReaders"/>
      <xsd:complexType name="AddReaders">
        <xsd:sequence>
          <xsd:element name="name" type="xsd:string"/>
          <xsd:element name="readers" minOccurs="0">
            <xsd:complexType>
              <xsd:sequence>
                <xsd:element name="reader" type="xsd:string" minOccurs="0"
                             maxOccurs="unbounded"/>
              </xsd:sequence>
            </xsd:complexType>
          </xsd:element>
        </xsd:sequence>
      </xsd:complexType>
      <xsd:element name="AddReadersResult">
        <xsd:complexType/>
      </xsd:element>

      <xsd:element name="SetReaders" type="impl:SetReaders"/>
      <xsd:complexType name="SetReaders">
        <xsd:sequence>
          <xsd:element name="name" type="xsd:string"/>
          <xsd:element name="readers" minOccurs="0">
            <xsd:complexType>
              <xsd:sequence>
                <xsd:element name="reader" type="xsd:string" minOccurs="0"
                             maxOccurs="unbounded"/>
              </xsd:sequence>
            </xsd:complexType>
          </xsd:element>
        </xsd:sequence>
      </xsd:complexType>
      <xsd:element name="SetReadersResult">
        <xsd:complexType/>
      </xsd:element>

      <xsd:element name="RemoveReaders" type="impl:RemoveReaders"/>
      <xsd:complexType name="RemoveReaders">
        <xsd:sequence>
          <xsd:element name="name" type="xsd:string"/>
          <xsd:element name="readers" minOccurs="0">
            <xsd:complexType>
              <xsd:sequence>
                <xsd:element name="reader" type="xsd:string" minOccurs="0"
                             maxOccurs="unbounded"/>
              </xsd:sequence>
            </xsd:complexType>
          </xsd:element>
        </xsd:sequence>
      </xsd:complexType>
      <xsd:element name="RemoveReadersResult">
        <xsd:complexType/>
      </xsd:element>

      <xsd:element name="SetProperties" type="impl:SetProperties"/>
      <xsd:complexType name="SetProperties">
        <xsd:sequence>
          <xsd:element name="name" type="xsd:string"/>
          <xsd:element name="properties" minOccurs="0">
            <xsd:complexType>
              <xsd:sequence>
                <xsd:element name="property" type="ale:LRProperty" minOccurs="0"
                             maxOccurs="unbounded"/>
              </xsd:sequence>
            </xsd:complexType>
          </xsd:element>
        </xsd:sequence>
      </xsd:complexType>
      <xsd:element name="SetPropertiesResult">
        <xsd:complexType/>
      </xsd:element>

      <xsd:element name="GetPropertyValue" type="impl:GetPropertyValue"/>
      <xsd:complexType name="GetPropertyValue">
        <xsd:sequence>
          <xsd:element name="name" type="xsd:string"/>
          <xsd:element name="propertyName" type="xsd:string"/>
        </xsd:sequence>
      </xsd:complexType>
```

```
        <xsd:element name="GetPropertyValueResult" type="xsd:string"/>

        <xsd:element name="GetStandardVersion" type="impl:EmptyParms"/>
        <xsd:element name="GetStandardVersionResult" type="xsd:string"/>

        <xsd:element name="GetVendorVersion" type="impl:EmptyParms"/>
        <xsd:element name="GetVendorVersionResult" type="xsd:string"/>

        <xsd:element name="ALEException" type="impl:ALEException"/>
        <xsd:complexType name="ALEException">
          <xsd:sequence>
            <xsd:element name="reason" type="xsd:string"/>
          </xsd:sequence>
        </xsd:complexType>

        <xsd:element name="DuplicateNameException" type="impl:DuplicateNameException"/>
        <xsd:complexType name="DuplicateNameException">
          <xsd:complexContent>
            <xsd:extension base="impl:ALEException"/>
          </xsd:complexContent>
        </xsd:complexType>

        <xsd:element name="NoSuchNameException" type="impl:NoSuchNameException"/>
        <xsd:complexType name="NoSuchNameException">
          <xsd:complexContent>
            <xsd:extension base="impl:ALEException"/>
          </xsd:complexContent>
        </xsd:complexType>

        <xsd:element name="InUseException" type="impl:InUseException"/>
        <xsd:complexType name="InUseException">
          <xsd:complexContent>
            <xsd:extension base="impl:ALEException"/>
          </xsd:complexContent>
        </xsd:complexType>

        <xsd:element name="ValidationException" type="impl:ValidationException"/>
        <xsd:complexType name="ValidationException">
          <xsd:complexContent>
            <xsd:extension base="impl:ALEException"/>
          </xsd:complexContent>
        </xsd:complexType>

        <xsd:element name="ImmutableReaderException" type="impl:ImmutableReaderException"/>
        <xsd:complexType name="ImmutableReaderException">
          <xsd:complexContent>
            <xsd:extension base="impl:ALEException"/>
          </xsd:complexContent>
        </xsd:complexType>

        <xsd:element name="NonCompositeReaderException"
                     type="impl:NonCompositeReaderException"/>
        <xsd:complexType name="NonCompositeReaderException">
          <xsd:complexContent>
            <xsd:extension base="impl:ALEException"/>
          </xsd:complexContent>
        </xsd:complexType>

        <xsd:element name="ReaderLoopException" type="impl:ReaderLoopException"/>
        <xsd:complexType name="ReaderLoopException">
          <xsd:complexContent>
            <xsd:extension base="impl:ALEException"/>
          </xsd:complexContent>
        </xsd:complexType>

        <xsd:element name="SecurityException" type="impl:SecurityException"/>
        <xsd:complexType name="SecurityException">
          <xsd:complexContent>
            <xsd:extension base="impl:ALEException"/>
          </xsd:complexContent>
        </xsd:complexType>

        <xsd:element name="ImplementationException" type="impl:ImplementationException"/>
        <xsd:complexType name="ImplementationException">
          <xsd:complexContent>
            <xsd:extension base="impl:ALEException">
              <xsd:sequence>
                <xsd:element name="severity" type="impl:ImplementationExceptionSeverity"/>
              </xsd:sequence>
            </xsd:extension>
          </xsd:complexContent>
        </xsd:complexType>

        <xsd:complexType name="ArrayOfString">
          <xsd:sequence>
            <xsd:element name="string" type="xsd:string" minOccurs="0"
                         maxOccurs="unbounded"/>
          </xsd:sequence>
        </xsd:complexType>

        <!-- The ImplementationExceptionSeverity type is an enumerated type.
             The following strings are legal values for this type:
               ERROR
               SEVERE
              -->
        <xsd:simpleType name="ImplementationExceptionSeverity">
          <xsd:restriction base="xsd:string"/>
        </xsd:simpleType>

        <xsd:complexType name="EmptyParms"/>
      </xsd:schema>
    </wsdl:types>
    <!-- ALELRSERVICE MESSAGES -->

    <wsdl:message name="defineRequest">
      <wsdl:part name="parms" element="impl:Define"/>
    </wsdl:message>
    <wsdl:message name="defineResponse">
      <wsdl:part name="defineReturn" element="impl:DefineResult"/>
    </wsdl:message>

    <wsdl:message name="updateRequest">
      <wsdl:part name="parms" element="impl:Update"/>
    </wsdl:message>
    <wsdl:message name="updateResponse">
      <wsdl:part name="updateReturn" element="impl:UpdateResult"/>
    </wsdl:message>

    <wsdl:message name="undefineRequest">
      <wsdl:part name="parms" element="impl:Undefine"/>
    </wsdl:message>
    <wsdl:message name="undefineResponse">
      <wsdl:part name="undefineReturn" element="impl:UndefineResult"/>
    </wsdl:message>

    <wsdl:message name="getLogicalReaderNamesRequest">
      <wsdl:part name="parms" element="impl:GetLogicalReaderNames"/>
    </wsdl:message>
    <wsdl:message name="getLogicalReaderNamesResponse">
      <wsdl:part name="getLogicalReaderNamesReturn"
                 element="impl:GetLogicalReaderNamesResult"/>
    </wsdl:message>

    <wsdl:message name="getLRSpecRequest">
      <wsdl:part name="parms" element="impl:GetLRSpec"/>
    </wsdl:message>
    <wsdl:message name="getLRSpecResponse">
      <wsdl:part name="getLRSpecReturn" element="impl:GetLRSpecResult"/>
    </wsdl:message>
```

```
  <wsdl:message name="addReadersRequest">
    <wsdl:part name="parms" element="impl:AddReaders"/>
  </wsdl:message>
  <wsdl:message name="addReadersResponse">
    <wsdl:part name="addReadersReturn" element="impl:AddReadersResult"/>
  </wsdl:message>

  <wsdl:message name="setReadersRequest">
    <wsdl:part name="parms" element="impl:SetReaders"/>
  </wsdl:message>
  <wsdl:message name="setReadersResponse">
    <wsdl:part name="setReadersReturn" element="impl:SetReadersResult"/>
  </wsdl:message>

  <wsdl:message name="removeReadersRequest">
    <wsdl:part name="parms" element="impl:RemoveReaders"/>
  </wsdl:message>
  <wsdl:message name="removeReadersResponse">
    <wsdl:part name="removeReadersReturn" element="impl:RemoveReadersResult"/>
  </wsdl:message>

  <wsdl:message name="setPropertiesRequest">
    <wsdl:part name="parms" element="impl:SetProperties"/>
  </wsdl:message>
  <wsdl:message name="setPropertiesResponse">
    <wsdl:part name="setPropertiesReturn" element="impl:SetPropertiesResult"/>
  </wsdl:message>

  <wsdl:message name="getPropertyValueRequest">
    <wsdl:part name="parms" element="impl:GetPropertyValue"/>
  </wsdl:message>
  <wsdl:message name="getPropertyValueResponse">
    <wsdl:part name="getPropertyValueReturn" element="impl:GetPropertyValueResult"/>
  </wsdl:message>

  <wsdl:message name="getStandardVersionRequest">
    <wsdl:part name="parms" element="impl:GetStandardVersion"/>
  </wsdl:message>
  <wsdl:message name="getStandardVersionResponse">
    <wsdl:part name="getStandardVersionReturn" element="impl:GetStandardVersionResult"/>
  </wsdl:message>

  <wsdl:message name="getVendorVersionRequest">
    <wsdl:part name="parms" element="impl:GetVendorVersion"/>
  </wsdl:message>
  <wsdl:message name="getVendorVersionResponse">
    <wsdl:part name="getVendorVersionReturn" element="impl:GetVendorVersionResult"/>
  </wsdl:message>

  <wsdl:message name="DuplicateNameExceptionResponse">
    <wsdl:part name="fault" element="impl:DuplicateNameException"/>
  </wsdl:message>

  <wsdl:message name="NoSuchNameExceptionResponse">
    <wsdl:part name="fault" element="impl:NoSuchNameException"/>
  </wsdl:message>

  <wsdl:message name="InUseExceptionResponse">
    <wsdl:part name="fault" element="impl:InUseException"/>
  </wsdl:message>

  <wsdl:message name="ValidationExceptionResponse">
    <wsdl:part name="fault" element="impl:ValidationException"/>
  </wsdl:message>

  <wsdl:message name="ImmutableReaderExceptionResponse">
    <wsdl:part name="fault" element="impl:ImmutableReaderException"/>
  </wsdl:message>

  <wsdl:message name="NonCompositeReaderExceptionResponse">
```

```
    <wsdl:part name="fault" element="impl:NonCompositeReaderException"/>
  </wsdl:message>

  <wsdl:message name="ReaderLoopExceptionResponse">
    <wsdl:part name="fault" element="impl:ReaderLoopException"/>
  </wsdl:message>

  <wsdl:message name="SecurityExceptionResponse">
    <wsdl:part name="fault" element="impl:SecurityException"/>
  </wsdl:message>

  <wsdl:message name="ImplementationExceptionResponse">
    <wsdl:part name="fault" element="impl:ImplementationException"/>
  </wsdl:message>
  <!-- ALELRSERVICE PORTTYPE -->

  <wsdl:portType name="ALELRServicePortType">

    <wsdl:operation name="define">
      <wsdl:input message="impl:defineRequest" name="defineRequest"/>
      <wsdl:output message="impl:defineResponse" name="defineResponse"/>
      <wsdl:fault message="impl:DuplicateNameExceptionResponse"
                  name="DuplicateNameExceptionFault"/>
      <wsdl:fault message="impl:ValidationExceptionResponse"
                  name="ValidationExceptionFault"/>
      <wsdl:fault message="impl:SecurityExceptionResponse"
                  name="SecurityExceptionFault"/>
      <wsdl:fault message="impl:ImplementationExceptionResponse"
                  name="ImplementationExceptionFault"/>
    </wsdl:operation>

    <wsdl:operation name="update">
      <wsdl:input message="impl:updateRequest" name="updateRequest"/>
      <wsdl:output message="impl:updateResponse" name="updateResponse"/>
      <wsdl:fault message="impl:NoSuchNameExceptionResponse"
                  name="NoSuchNameExceptionFault"/>
      <wsdl:fault message="impl:ValidationExceptionResponse"
                  name="ValidationExceptionFault"/>
      <wsdl:fault message="impl:InUseExceptionResponse" name="InUseExceptionFault"/>
      <wsdl:fault message="impl:ImmutableReaderExceptionResponse"
                  name="ImmutableReaderExceptionFault"/>
      <wsdl:fault message="impl:ReaderLoopExceptionResponse"
                  name="ReaderLoopExceptionFault"/>
      <wsdl:fault message="impl:SecurityExceptionResponse"
                  name="SecurityExceptionFault"/>
      <wsdl:fault message="impl:ImplementationExceptionResponse"
                  name="ImplementationExceptionFault"/>
    </wsdl:operation>

    <wsdl:operation name="undefine">
      <wsdl:input message="impl:undefineRequest" name="undefineRequest"/>
      <wsdl:output message="impl:undefineResponse" name="undefineResponse"/>
      <wsdl:fault message="impl:NoSuchNameExceptionResponse"
                  name="NoSuchNameExceptionFault"/>
      <wsdl:fault message="impl:InUseExceptionResponse" name="InUseExceptionFault"/>
      <wsdl:fault message="impl:ImmutableReaderExceptionResponse"
                  name="ImmutableReaderExceptionFault"/>
      <wsdl:fault message="impl:SecurityExceptionResponse"
                  name="SecurityExceptionFault"/>
      <wsdl:fault message="impl:ImplementationExceptionResponse"
                  name="ImplementationExceptionFault"/>
    </wsdl:operation>

    <wsdl:operation name="getLogicalReaderNames">
      <wsdl:input message="impl:getLogicalReaderNamesRequest"
                  name="getLogicalReaderNamesRequest"/>
      <wsdl:output message="impl:getLogicalReaderNamesResponse"
                   name="getLogicalReaderNamesResponse"/>
      <wsdl:fault message="impl:SecurityExceptionResponse"
                  name="SecurityExceptionFault"/>
      <wsdl:fault message="impl:ImplementationExceptionResponse"
                  name="ImplementationExceptionFault"/>
    </wsdl:operation>

    <wsdl:operation name="getLRSpec">
      <wsdl:input message="impl:getLRSpecRequest" name="getLRSpecRequest"/>
      <wsdl:output message="impl:getLRSpecResponse" name="getLRSpecResponse"/>
      <wsdl:fault message="impl:NoSuchNameExceptionResponse"
                  name="NoSuchNameExceptionFault"/>
      <wsdl:fault message="impl:SecurityExceptionResponse"
                  name="SecurityExceptionFault"/>
      <wsdl:fault message="impl:ImplementationExceptionResponse"
                  name="ImplementationExceptionFault"/>
    </wsdl:operation>

    <wsdl:operation name="addReaders">
      <wsdl:input message="impl:addReadersRequest" name="addReadersRequest"/>
      <wsdl:output message="impl:addReadersResponse" name="addReadersResponse"/>
      <wsdl:fault message="impl:NoSuchNameExceptionResponse"
                  name="NoSuchNameExceptionFault"/>
      <wsdl:fault message="impl:ValidationExceptionResponse"
                  name="ValidationExceptionFault"/>
      <wsdl:fault message="impl:InUseExceptionResponse" name="InUseExceptionFault"/>
      <wsdl:fault message="impl:ImmutableReaderExceptionResponse"
                  name="ImmutableReaderExceptionFault"/>
      <wsdl:fault message="impl:NonCompositeReaderExceptionResponse"
                  name="NonCompositeReaderExceptionFault"/>
      <wsdl:fault message="impl:ReaderLoopExceptionResponse"
                  name="ReaderLoopExceptionFault"/>
      <wsdl:fault message="impl:SecurityExceptionResponse"
                  name="SecurityExceptionFault"/>
      <wsdl:fault message="impl:ImplementationExceptionResponse"
                  name="ImplementationExceptionFault"/>
    </wsdl:operation>

    <wsdl:operation name="setReaders">
      <wsdl:input message="impl:setReadersRequest" name="setReadersRequest"/>
      <wsdl:output message="impl:setReadersResponse" name="setReadersResponse"/>
      <wsdl:fault message="impl:NoSuchNameExceptionResponse"
                  name="NoSuchNameExceptionFault"/>
      <wsdl:fault message="impl:ValidationExceptionResponse"
                  name="ValidationExceptionFault"/>
      <wsdl:fault message="impl:InUseExceptionResponse" name="InUseExceptionFault"/>
      <wsdl:fault message="impl:ImmutableReaderExceptionResponse"
                  name="ImmutableReaderExceptionFault"/>
      <wsdl:fault message="impl:NonCompositeReaderExceptionResponse"
                  name="NonCompositeReaderExceptionFault"/>
      <wsdl:fault message="impl:ReaderLoopExceptionResponse"
                  name="ReaderLoopExceptionFault"/>
      <wsdl:fault message="impl:SecurityExceptionResponse"
                  name="SecurityExceptionFault"/>
      <wsdl:fault message="impl:ImplementationExceptionResponse"
                  name="ImplementationExceptionFault"/>
    </wsdl:operation>

    <wsdl:operation name="removeReaders">
      <wsdl:input message="impl:removeReadersRequest" name="removeReadersRequest"/>
      <wsdl:output message="impl:removeReadersResponse" name="removeReadersResponse"/>
      <wsdl:fault message="impl:NoSuchNameExceptionResponse"
                  name="NoSuchNameExceptionFault"/>
      <wsdl:fault message="impl:InUseExceptionResponse" name="InUseExceptionFault"/>
      <wsdl:fault message="impl:ImmutableReaderExceptionResponse"
                  name="ImmutableReaderExceptionFault"/>
      <wsdl:fault message="impl:NonCompositeReaderExceptionResponse"
                  name="NonCompositeReaderExceptionFault"/>
      <wsdl:fault message="impl:SecurityExceptionResponse"
                  name="SecurityExceptionFault"/>
      <wsdl:fault message="impl:ImplementationExceptionResponse"
                  name="ImplementationExceptionFault"/>
    </wsdl:operation>
```

```

    <wsdl:operation name="setProperties">
      <wsdl:input message="impl:setPropertiesRequest" name="setPropertiesRequest"/>
      <wsdl:output message="impl:setPropertiesResponse" name="setPropertiesResponse"/>
      <wsdl:fault message="impl:NoSuchNameExceptionResponse"
                  name="NoSuchNameExceptionFault"/>
      <wsdl:fault message="impl:ValidationExceptionResponse"
                  name="ValidationExceptionFault"/>
      <wsdl:fault message="impl:InUseExceptionResponse" name="InUseExceptionFault"/>
      <wsdl:fault message="impl:ImmutableReaderExceptionResponse"
                  name="ImmutableReaderExceptionFault"/>
      <wsdl:fault message="impl:SecurityExceptionResponse"
                  name="SecurityExceptionFault"/>
      <wsdl:fault message="impl:ImplementationExceptionResponse"
                  name="ImplementationExceptionFault"/>
    </wsdl:operation>

    <wsdl:operation name="getPropertyValue">
      <wsdl:input message="impl:getPropertyValueRequest" name="getPropertyValueRequest"/>
      <wsdl:output message="impl:getPropertyValueResponse"
                   name="getPropertyValueResponse"/>
      <wsdl:fault message="impl:NoSuchNameExceptionResponse"
                  name="NoSuchNameExceptionFault"/>
      <wsdl:fault message="impl:SecurityExceptionResponse"
                  name="SecurityExceptionFault"/>
      <wsdl:fault message="impl:ImplementationExceptionResponse"
                  name="ImplementationExceptionFault"/>
    </wsdl:operation>

    <wsdl:operation name="getStandardVersion">
      <wsdl:input message="impl:getStandardVersionRequest"
                  name="getStandardVersionRequest"/>
      <wsdl:output message="impl:getStandardVersionResponse"
                   name="getStandardVersionResponse"/>
      <wsdl:fault message="impl:ImplementationExceptionResponse"
                  name="ImplementationExceptionFault"/>
    </wsdl:operation>

    <wsdl:operation name="getVendorVersion">
      <wsdl:input message="impl:getVendorVersionRequest" name="getVendorVersionRequest"/>
      <wsdl:output message="impl:getVendorVersionResponse"
                   name="getVendorVersionResponse"/>
      <wsdl:fault message="impl:ImplementationExceptionResponse"
                  name="ImplementationExceptionFault"/>
    </wsdl:operation>
  </wsdl:portType>
  <!-- ALELRSERVICE BINDING -->

  <wsdl:binding name="ALELRServiceBinding" type="impl:ALELRServicePortType">
    <wsdlsoap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>

    <wsdl:operation name="define">
      <wsdlsoap:operation soapAction=""/>
      <wsdl:input name="defineRequest">
        <wsdlsoap:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="defineResponse">
        <wsdlsoap:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="DuplicateNameExceptionFault">
        <wsdlsoap:fault name="DuplicateNameExceptionFault" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="ValidationExceptionFault">
        <wsdlsoap:fault name="ValidationExceptionFault" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="SecurityExceptionFault">
        <wsdlsoap:fault name="SecurityExceptionFault" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="ImplementationExceptionFault">
        <wsdlsoap:fault name="ImplementationExceptionFault" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>

    <wsdl:operation name="update">
      <wsdlsoap:operation soapAction=""/>
      <wsdl:input name="updateRequest">
        <wsdlsoap:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="updateResponse">
        <wsdlsoap:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="NoSuchNameExceptionFault">
        <wsdlsoap:fault name="NoSuchNameExceptionFault" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="ValidationExceptionFault">
        <wsdlsoap:fault name="ValidationExceptionFault" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="InUseExceptionFault">
        <wsdlsoap:fault name="InUseExceptionFault" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="ImmutableReaderExceptionFault">
        <wsdlsoap:fault name="ImmutableReaderExceptionFault" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="ReaderLoopExceptionFault">
        <wsdlsoap:fault name="ReaderLoopExceptionFault" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="SecurityExceptionFault">
        <wsdlsoap:fault name="SecurityExceptionFault" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="ImplementationExceptionFault">
        <wsdlsoap:fault name="ImplementationExceptionFault" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>

    <wsdl:operation name="undefine">
      <wsdlsoap:operation soapAction=""/>
      <wsdl:input name="undefineRequest">
        <wsdlsoap:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="undefineResponse">
        <wsdlsoap:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="NoSuchNameExceptionFault">
        <wsdlsoap:fault name="NoSuchNameExceptionFault" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="InUseExceptionFault">
        <wsdlsoap:fault name="InUseExceptionFault" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="ImmutableReaderExceptionFault">
        <wsdlsoap:fault name="ImmutableReaderExceptionFault" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="SecurityExceptionFault">
        <wsdlsoap:fault name="SecurityExceptionFault" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="ImplementationExceptionFault">
        <wsdlsoap:fault name="ImplementationExceptionFault" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>

    <wsdl:operation name="getLogicalReaderNames">
      <wsdlsoap:operation soapAction=""/>
      <wsdl:input name="getLogicalReaderNamesRequest">
        <wsdlsoap:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="getLogicalReaderNamesResponse">
        <wsdlsoap:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="SecurityExceptionFault">
        <wsdlsoap:fault name="SecurityExceptionFault" use="literal"/>
      </wsdl:fault>
```

```
          <wsdl:fault name="ImplementationExceptionFault">
            <wsdlsoap:fault name="ImplementationExceptionFault" use="literal"/>
          </wsdl:fault>
        </wsdl:operation>

        <wsdl:operation name="getLRSpec">
          <wsdlsoap:operation soapAction=""/>
          <wsdl:input name="getLRSpecRequest">
            <wsdlsoap:body use="literal"/>
          </wsdl:input>
          <wsdl:output name="getLRSpecResponse">
            <wsdlsoap:body use="literal"/>
          </wsdl:output>
          <wsdl:fault name="NoSuchNameExceptionFault">
            <wsdlsoap:fault name="NoSuchNameExceptionFault" use="literal"/>
          </wsdl:fault>
          <wsdl:fault name="SecurityExceptionFault">
            <wsdlsoap:fault name="SecurityExceptionFault" use="literal"/>
          </wsdl:fault>
          <wsdl:fault name="ImplementationExceptionFault">
            <wsdlsoap:fault name="ImplementationExceptionFault" use="literal"/>
          </wsdl:fault>
        </wsdl:operation>

        <wsdl:operation name="addReaders">
          <wsdlsoap:operation soapAction=""/>
          <wsdl:input name="addReadersRequest">
            <wsdlsoap:body use="literal"/>
          </wsdl:input>
          <wsdl:output name="addReadersResponse">
            <wsdlsoap:body use="literal"/>
          </wsdl:output>
          <wsdl:fault name="NoSuchNameExceptionFault">
            <wsdlsoap:fault name="NoSuchNameExceptionFault" use="literal"/>
          </wsdl:fault>
          <wsdl:fault name="ValidationExceptionFault">
            <wsdlsoap:fault name="ValidationExceptionFault" use="literal"/>
          </wsdl:fault>
          <wsdl:fault name="InUseExceptionFault">
            <wsdlsoap:fault name="InUseExceptionFault" use="literal"/>
          </wsdl:fault>
          <wsdl:fault name="ImmutableReaderExceptionFault">
            <wsdlsoap:fault name="ImmutableReaderExceptionFault" use="literal"/>
          </wsdl:fault>
          <wsdl:fault name="NonCompositeReaderExceptionFault">
            <wsdlsoap:fault name="NonCompositeReaderExceptionFault" use="literal"/>
          </wsdl:fault>
          <wsdl:fault name="ReaderLoopExceptionFault">
            <wsdlsoap:fault name="ReaderLoopExceptionFault" use="literal"/>
          </wsdl:fault>
          <wsdl:fault name="SecurityExceptionFault">
            <wsdlsoap:fault name="SecurityExceptionFault" use="literal"/>
          </wsdl:fault>
          <wsdl:fault name="ImplementationExceptionFault">
            <wsdlsoap:fault name="ImplementationExceptionFault" use="literal"/>
          </wsdl:fault>
        </wsdl:operation>

        <wsdl:operation name="setReaders">
          <wsdlsoap:operation soapAction=""/>
          <wsdl:input name="setReadersRequest">
            <wsdlsoap:body use="literal"/>
          </wsdl:input>
          <wsdl:output name="setReadersResponse">
            <wsdlsoap:body use="literal"/>
          </wsdl:output>
          <wsdl:fault name="NoSuchNameExceptionFault">
            <wsdlsoap:fault name="NoSuchNameExceptionFault" use="literal"/>
          </wsdl:fault>
          <wsdl:fault name="ValidationExceptionFault">
            <wsdlsoap:fault name="ValidationExceptionFault" use="literal"/>
          </wsdl:fault>
          <wsdl:fault name="InUseExceptionFault">
            <wsdlsoap:fault name="InUseExceptionFault" use="literal"/>
          </wsdl:fault>
          <wsdl:fault name="ImmutableReaderExceptionFault">
            <wsdlsoap:fault name="ImmutableReaderExceptionFault" use="literal"/>
          </wsdl:fault>
          <wsdl:fault name="NonCompositeReaderExceptionFault">
            <wsdlsoap:fault name="NonCompositeReaderExceptionFault" use="literal"/>
          </wsdl:fault>
          <wsdl:fault name="ReaderLoopExceptionFault">
            <wsdlsoap:fault name="ReaderLoopExceptionFault" use="literal"/>
          </wsdl:fault>
          <wsdl:fault name="SecurityExceptionFault">
            <wsdlsoap:fault name="SecurityExceptionFault" use="literal"/>
          </wsdl:fault>
          <wsdl:fault name="ImplementationExceptionFault">
            <wsdlsoap:fault name="ImplementationExceptionFault" use="literal"/>
          </wsdl:fault>
        </wsdl:operation>

        <wsdl:operation name="removeReaders">
          <wsdlsoap:operation soapAction=""/>
          <wsdl:input name="removeReadersRequest">
            <wsdlsoap:body use="literal"/>
          </wsdl:input>
          <wsdl:output name="removeReadersResponse">
            <wsdlsoap:body use="literal"/>
          </wsdl:output>
          <wsdl:fault name="NoSuchNameExceptionFault">
            <wsdlsoap:fault name="NoSuchNameExceptionFault" use="literal"/>
          </wsdl:fault>
          <wsdl:fault name="InUseExceptionFault">
            <wsdlsoap:fault name="InUseExceptionFault" use="literal"/>
          </wsdl:fault>
          <wsdl:fault name="ImmutableReaderExceptionFault">
            <wsdlsoap:fault name="ImmutableReaderExceptionFault" use="literal"/>
          </wsdl:fault>
          <wsdl:fault name="NonCompositeReaderExceptionFault">
            <wsdlsoap:fault name="NonCompositeReaderExceptionFault" use="literal"/>
          </wsdl:fault>
          <wsdl:fault name="SecurityExceptionFault">
            <wsdlsoap:fault name="SecurityExceptionFault" use="literal"/>
          </wsdl:fault>
          <wsdl:fault name="ImplementationExceptionFault">
            <wsdlsoap:fault name="ImplementationExceptionFault" use="literal"/>
          </wsdl:fault>
        </wsdl:operation>

        <wsdl:operation name="setProperties">
          <wsdlsoap:operation soapAction=""/>
          <wsdl:input name="setPropertiesRequest">
            <wsdlsoap:body use="literal"/>
          </wsdl:input>
          <wsdl:output name="setPropertiesResponse">
            <wsdlsoap:body use="literal"/>
          </wsdl:output>
          <wsdl:fault name="NoSuchNameExceptionFault">
            <wsdlsoap:fault name="NoSuchNameExceptionFault" use="literal"/>
          </wsdl:fault>
          <wsdl:fault name="ValidationExceptionFault">
            <wsdlsoap:fault name="ValidationExceptionFault" use="literal"/>
          </wsdl:fault>
          <wsdl:fault name="InUseExceptionFault">
            <wsdlsoap:fault name="InUseExceptionFault" use="literal"/>
          </wsdl:fault>
          <wsdl:fault name="ImmutableReaderExceptionFault">
            <wsdlsoap:fault name="ImmutableReaderExceptionFault" use="literal"/>
          </wsdl:fault>
```

```
        <wsdl:fault name="SecurityExceptionFault">
          <wsdlsoap:fault name="SecurityExceptionFault" use="literal"/>
        </wsdl:fault>
        <wsdl:fault name="ImplementationExceptionFault">
          <wsdlsoap:fault name="ImplementationExceptionFault" use="literal"/>
        </wsdl:fault>
      </wsdl:operation>

      <wsdl:operation name="getPropertyValue">
        <wsdlsoap:operation soapAction=""/>
        <wsdl:input name="getPropertyValueRequest">
          <wsdlsoap:body use="literal"/>
        </wsdl:input>
        <wsdl:output name="getPropertyValueResponse">
          <wsdlsoap:body use="literal"/>
        </wsdl:output>
        <wsdl:fault name="NoSuchNameExceptionFault">
          <wsdlsoap:fault name="NoSuchNameExceptionFault" use="literal"/>
        </wsdl:fault>
        <wsdl:fault name="SecurityExceptionFault">
          <wsdlsoap:fault name="SecurityExceptionFault" use="literal"/>
        </wsdl:fault>
        <wsdl:fault name="ImplementationExceptionFault">
          <wsdlsoap:fault name="ImplementationExceptionFault" use="literal"/>
        </wsdl:fault>
      </wsdl:operation>

      <wsdl:operation name="getStandardVersion">
        <wsdlsoap:operation soapAction=""/>
        <wsdl:input name="getStandardVersionRequest">
          <wsdlsoap:body use="literal"/>
        </wsdl:input>
        <wsdl:output name="getStandardVersionResponse">
          <wsdlsoap:body use="literal"/>
        </wsdl:output>
        <wsdl:fault name="ImplementationExceptionFault">
          <wsdlsoap:fault name="ImplementationExceptionFault" use="literal"/>
        </wsdl:fault>
      </wsdl:operation>

      <wsdl:operation name="getVendorVersion">
        <wsdlsoap:operation soapAction=""/>
        <wsdl:input name="getVendorVersionRequest">
          <wsdlsoap:body use="literal"/>
        </wsdl:input>
        <wsdl:output name="getVendorVersionResponse">
          <wsdlsoap:body use="literal"/>
        </wsdl:output>
        <wsdl:fault name="ImplementationExceptionFault">
          <wsdlsoap:fault name="ImplementationExceptionFault" use="literal"/>
        </wsdl:fault>
      </wsdl:operation>
    </wsdl:binding>

    <!-- ALELRSERVICE -->
    <wsdl:service name="ALELRService">
      <wsdl:port binding="impl:ALELRServiceBinding" name="ALELRServicePort">
        <!-- The value of the location attribute below is an example only;
             Implementations are free to choose any appropriate URL. -->
        <wsdlsoap:address location="http://localhost:8080/services/ALELRService"/>
      </wsdl:port>
    </wsdl:service>
  </wsdl:definitions>
```

4.7 ALE Access Control API SOAP Binding

The following is a Web Services Description Language (WSDL) 1.1 [WSDL1.1] specification defining the standard SOAP 1.1 [SOAP1.1] binding of the ALE Access

Control API. This SOAP binding is compliant with the WS-I Basic Profile Version 1.0 [WSI].

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- ALEACSERVICE DEFINITIONS -->
<wsdl:definitions xmlns:xsd="http://www.w3.org/2001/XMLSchema"
                  xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
                  xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/"
                  xmlns:ale="urn:epcglobal:ale:xsd:1"
                  xmlns:impl="urn:epcglobal:aleac:wsdl:1"
                  targetNamespace="urn:epcglobal:aleac:wsdl:1">
  <!-- ALEACSERVICE TYPES -->
  <wsdl:types>
    <xsd:schema targetNamespace="urn:epcglobal:aleac:wsdl:1">
      <xsd:import namespace="urn:epcglobal:ale:xsd:1"
                  schemaLocation="EPCglobal-ale-1_1-aleac.xsd"/>
      <!-- ALEACSERVICE MESSAGE WRAPPERS -->

      <xsd:element name="GetPermissionNames" type="impl:EmptyParms"/>
      <xsd:element name="GetPermissionNamesResult" type="impl:ArrayOfString"/>

      <xsd:element name="DefinePermission" type="impl:DefinePermission"/>
      <xsd:complexType name="DefinePermission">
        <xsd:sequence>
          <xsd:element name="permName" type="xsd:string"/>
          <xsd:element name="perm" type="ale:ACPermission"/>
        </xsd:sequence>
      </xsd:complexType>
      <xsd:element name="DefinePermissionResult">
        <xsd:complexType/>
      </xsd:element>

      <xsd:element name="UpdatePermission" type="impl:UpdatePermission"/>
      <xsd:complexType name="UpdatePermission">
        <xsd:sequence>
          <xsd:element name="permName" type="xsd:string"/>
          <xsd:element name="perm" type="ale:ACPermission"/>
        </xsd:sequence>
      </xsd:complexType>
      <xsd:element name="UpdatePermissionResult">
        <xsd:complexType/>
      </xsd:element>

      <xsd:element name="GetPermission" type="impl:GetPermission"/>
      <xsd:complexType name="GetPermission">
        <xsd:sequence>
          <xsd:element name="permName" type="xsd:string"/>
        </xsd:sequence>
      </xsd:complexType>
      <xsd:element name="GetPermissionResult" type="ale:ACPermission"/>

      <xsd:element name="UndefinePermission" type="impl:UndefinePermission"/>
      <xsd:complexType name="UndefinePermission">
        <xsd:sequence>
          <xsd:element name="permName" type="xsd:string"/>
        </xsd:sequence>
      </xsd:complexType>
      <xsd:element name="UndefinePermissionResult">
        <xsd:complexType/>
      </xsd:element>

      <xsd:element name="GetRoleNames" type="impl:EmptyParms"/>
      <xsd:element name="GetRoleNamesResult" type="impl:ArrayOfString"/>

      <xsd:element name="DefineRole" type="impl:DefineRole"/>
      <xsd:complexType name="DefineRole">
        <xsd:sequence>
          <xsd:element name="roleName" type="xsd:string"/>
          <xsd:element name="role" type="ale:ACRole"/>
        </xsd:sequence>
      </xsd:complexType>
      <xsd:element name="DefineRoleResult">
        <xsd:complexType/>
      </xsd:element>

      <xsd:element name="UpdateRole" type="impl:UpdateRole"/>
      <xsd:complexType name="UpdateRole">
        <xsd:sequence>
          <xsd:element name="roleName" type="xsd:string"/>
          <xsd:element name="role" type="ale:ACRole"/>
        </xsd:sequence>
      </xsd:complexType>
      <xsd:element name="UpdateRoleResult">
        <xsd:complexType/>
      </xsd:element>

      <xsd:element name="GetRole" type="impl:GetRole"/>
      <xsd:complexType name="GetRole">
        <xsd:sequence>
          <xsd:element name="roleName" type="xsd:string"/>
        </xsd:sequence>
      </xsd:complexType>
      <xsd:element name="GetRoleResult" type="ale:ACRole"/>

      <xsd:element name="UndefineRole" type="impl:UndefineRole"/>
      <xsd:complexType name="UndefineRole">
        <xsd:sequence>
          <xsd:element name="roleName" type="xsd:string"/>
        </xsd:sequence>
      </xsd:complexType>
      <xsd:element name="UndefineRoleResult">
        <xsd:complexType/>
      </xsd:element>

      <xsd:element name="AddPermissions" type="impl:AddPermissions"/>
      <xsd:complexType name="AddPermissions">
        <xsd:sequence>
          <xsd:element name="roleName" type="xsd:string"/>
          <xsd:element name="permissionNames" minOccurs="0">
            <xsd:complexType>
              <xsd:sequence>
                <xsd:element name="permissionName" type="xsd:string" minOccurs="0"
                             maxOccurs="unbounded"/>
              </xsd:sequence>
            </xsd:complexType>
          </xsd:element>
        </xsd:sequence>
      </xsd:complexType>
      <xsd:element name="AddPermissionsResult">
        <xsd:complexType/>
      </xsd:element>

      <xsd:element name="SetPermissions" type="impl:SetPermissions"/>
      <xsd:complexType name="SetPermissions">
        <xsd:sequence>
          <xsd:element name="roleName" type="xsd:string"/>
          <xsd:element name="permissionNames" minOccurs="0">
            <xsd:complexType>
              <xsd:sequence>
                <xsd:element name="permissionName" type="xsd:string" minOccurs="0"
                             maxOccurs="unbounded"/>
              </xsd:sequence>
            </xsd:complexType>
          </xsd:element>
        </xsd:sequence>
      </xsd:complexType>
      <xsd:element name="SetPermissionsResult">
        <xsd:complexType/>
      </xsd:element>

```

```
    <xsd:element name="RemovePermissions" type="impl:RemovePermissions"/>
    <xsd:complexType name="RemovePermissions">
      <xsd:sequence>
        <xsd:element name="roleName" type="xsd:string"/>
        <xsd:element name="permissionNames" minOccurs="0">
          <xsd:complexType>
            <xsd:sequence>
              <xsd:element name="permissionName" type="xsd:string" minOccurs="0"
                           maxOccurs="unbounded"/>
            </xsd:sequence>
          </xsd:complexType>
        </xsd:element>
      </xsd:sequence>
    </xsd:complexType>
    <xsd:element name="RemovePermissionsResult">
      <xsd:complexType/>
    </xsd:element>

    <xsd:element name="GetClientIdentityNames" type="impl:EmptyParms"/>
    <xsd:element name="GetClientIdentityNamesResult" type="impl:ArrayOfString"/>

    <xsd:element name="DefineClientIdentity" type="impl:DefineClientIdentity"/>
    <xsd:complexType name="DefineClientIdentity">
      <xsd:sequence>
        <xsd:element name="identityName" type="xsd:string"/>
        <xsd:element name="id" type="ale:ACClientIdentity"/>
      </xsd:sequence>
    </xsd:complexType>
    <xsd:element name="DefineClientIdentityResult">
      <xsd:complexType/>
    </xsd:element>

    <xsd:element name="UpdateClientIdentity" type="impl:UpdateClientIdentity"/>
    <xsd:complexType name="UpdateClientIdentity">
      <xsd:sequence>
        <xsd:element name="identityName" type="xsd:string"/>
        <xsd:element name="id" type="ale:ACClientIdentity"/>
      </xsd:sequence>
    </xsd:complexType>
    <xsd:element name="UpdateClientIdentityResult">
      <xsd:complexType/>
    </xsd:element>

    <xsd:element name="GetClientIdentity" type="impl:GetClientIdentity"/>
    <xsd:complexType name="GetClientIdentity">
      <xsd:sequence>
        <xsd:element name="identityName" type="xsd:string"/>
      </xsd:sequence>
    </xsd:complexType>
    <xsd:element name="GetClientIdentityResult" type="ale:ACClientIdentity"/>

    <xsd:element name="GetClientPermissionNames" type="impl:GetClientPermissionNames"/>
    <xsd:complexType name="GetClientPermissionNames">
      <xsd:sequence>
        <xsd:element name="identityName" type="xsd:string"/>
      </xsd:sequence>
    </xsd:complexType>
    <xsd:element name="GetClientPermissionNamesResult" type="impl:ArrayOfString"/>

    <xsd:element name="UndefineClientIdentity" type="impl:UndefineClientIdentity"/>
    <xsd:complexType name="UndefineClientIdentity">
      <xsd:sequence>
        <xsd:element name="identityName" type="xsd:string"/>
      </xsd:sequence>
    </xsd:complexType>
    <xsd:element name="UndefineClientIdentityResult">
      <xsd:complexType/>
    </xsd:element>

    <xsd:element name="AddRoles" type="impl:AddRoles"/>
```

```
<xsd:complexType name="AddRoles">
  <xsd:sequence>
    <xsd:element name="identityName" type="xsd:string"/>
    <xsd:element name="roleNames" minOccurs="0">
      <xsd:complexType>
        <xsd:sequence>
          <xsd:element name="roleName" type="xsd:string" minOccurs="0"
                       maxOccurs="unbounded"/>
        </xsd:sequence>
      </xsd:complexType>
    </xsd:element>
  </xsd:sequence>
</xsd:complexType>
<xsd:element name="AddRolesResult">
  <xsd:complexType/>
</xsd:element>

<xsd:element name="RemoveRoles" type="impl:RemoveRoles"/>
<xsd:complexType name="RemoveRoles">
  <xsd:sequence>
    <xsd:element name="identityName" type="xsd:string"/>
    <xsd:element name="roleNames" minOccurs="0">
      <xsd:complexType>
        <xsd:sequence>
          <xsd:element name="roleName" type="xsd:string" minOccurs="0"
                       maxOccurs="unbounded"/>
        </xsd:sequence>
      </xsd:complexType>
    </xsd:element>
  </xsd:sequence>
</xsd:complexType>
<xsd:element name="RemoveRolesResult">
  <xsd:complexType/>
</xsd:element>

<xsd:element name="SetRoles" type="impl:SetRoles"/>
<xsd:complexType name="SetRoles">
  <xsd:sequence>
    <xsd:element name="identityName" type="xsd:string"/>
    <xsd:element name="roleNames" minOccurs="0">
      <xsd:complexType>
        <xsd:sequence>
          <xsd:element name="roleName" type="xsd:string" minOccurs="0"
                       maxOccurs="unbounded"/>
        </xsd:sequence>
      </xsd:complexType>
    </xsd:element>
  </xsd:sequence>
</xsd:complexType>
<xsd:element name="SetRolesResult">
  <xsd:complexType/>
</xsd:element>

<xsd:element name="GetSupportedOperations" type="impl:EmptyParms"/>
<xsd:element name="GetSupportedOperationsResult" type="impl:ArrayOfString"/>

<xsd:element name="GetStandardVersion" type="impl:EmptyParms"/>
<xsd:element name="GetStandardVersionResult" type="xsd:string"/>

<xsd:element name="GetVendorVersion" type="impl:EmptyParms"/>
<xsd:element name="GetVendorVersionResult" type="xsd:string"/>

<xsd:element name="ALEException" type="impl:ALEException"/>
<xsd:complexType name="ALEException">
  <xsd:sequence>
    <xsd:element name="reason" type="xsd:string"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:element name="SecurityException" type="impl:SecurityException"/>
```

```
<xsd:complexType name="SecurityException">
  <xsd:complexContent>
    <xsd:extension base="impl:ALEException"/>
  </xsd:complexContent>
</xsd:complexType>

<xsd:element name="NoSuchPermissionException"
             type="impl:NoSuchPermissionException"/>
<xsd:complexType name="NoSuchPermissionException">
  <xsd:complexContent>
    <xsd:extension base="impl:ALEException"/>
  </xsd:complexContent>
</xsd:complexType>

<xsd:element name="PermissionValidationException"
             type="impl:PermissionValidationException"/>
<xsd:complexType name="PermissionValidationException">
  <xsd:complexContent>
    <xsd:extension base="impl:ALEException"/>
  </xsd:complexContent>
</xsd:complexType>

<xsd:element name="DuplicatePermissionException"
             type="impl:DuplicatePermissionException"/>
<xsd:complexType name="DuplicatePermissionException">
  <xsd:complexContent>
    <xsd:extension base="impl:ALEException"/>
  </xsd:complexContent>
</xsd:complexType>

<xsd:element name="NoSuchRoleException" type="impl:NoSuchRoleException"/>
<xsd:complexType name="NoSuchRoleException">
  <xsd:complexContent>
    <xsd:extension base="impl:ALEException"/>
  </xsd:complexContent>
</xsd:complexType>

<xsd:element name="RoleValidationException" type="impl:RoleValidationException"/>
<xsd:complexType name="RoleValidationException">
  <xsd:complexContent>
    <xsd:extension base="impl:ALEException"/>
  </xsd:complexContent>
</xsd:complexType>

<xsd:element name="DuplicateRoleException" type="impl:DuplicateRoleException"/>
<xsd:complexType name="DuplicateRoleException">
  <xsd:complexContent>
    <xsd:extension base="impl:ALEException"/>
  </xsd:complexContent>
</xsd:complexType>

<xsd:element name="NoSuchClientIdentityException"
             type="impl:NoSuchClientIdentityException"/>
<xsd:complexType name="NoSuchClientIdentityException">
  <xsd:complexContent>
    <xsd:extension base="impl:ALEException"/>
  </xsd:complexContent>
</xsd:complexType>

<xsd:element name="ClientIdentityValidationException"
             type="impl:ClientIdentityValidationException"/>
<xsd:complexType name="ClientIdentityValidationException">
  <xsd:complexContent>
    <xsd:extension base="impl:ALEException"/>
  </xsd:complexContent>
</xsd:complexType>

<xsd:element name="DuplicateClientIdentityException"
             type="impl:DuplicateClientIdentityException"/>
<xsd:complexType name="DuplicateClientIdentityException">
```

```
        <xsd:complexContent>
          <xsd:extension base="impl:ALEException"/>
        </xsd:complexContent>
      </xsd:complexType>

      <xsd:element name="UnsupportedOperationException"
                   type="impl:UnsupportedOperationException"/>
      <xsd:complexType name="UnsupportedOperationException">
        <xsd:complexContent>
          <xsd:extension base="impl:ALEException"/>
        </xsd:complexContent>
      </xsd:complexType>

      <xsd:element name="ImplementationException" type="impl:ImplementationException"/>
      <xsd:complexType name="ImplementationException">
        <xsd:complexContent>
          <xsd:extension base="impl:ALEException">
            <xsd:sequence>
              <xsd:element name="severity" type="impl:ImplementationExceptionSeverity"/>
            </xsd:sequence>
          </xsd:extension>
        </xsd:complexContent>
      </xsd:complexType>

      <xsd:complexType name="ArrayOfString">
        <xsd:sequence>
          <xsd:element name="string" type="xsd:string" minOccurs="0"
                       maxOccurs="unbounded"/>
        </xsd:sequence>
      </xsd:complexType>

      <!-- The ImplementationExceptionSeverity type is an enumerated type.
           The following strings are legal values for this type:
             ERROR
             SEVERE
            -->
      <xsd:simpleType name="ImplementationExceptionSeverity">
        <xsd:restriction base="xsd:string"/>
      </xsd:simpleType>

      <xsd:complexType name="EmptyParms"/>
    </xsd:schema>
  </wsdl:types>
  <!-- ALEACSERVICE MESSAGES -->

  <wsdl:message name="getPermissionNamesRequest">
    <wsdl:part name="parms" element="impl:GetPermissionNames"/>
  </wsdl:message>
  <wsdl:message name="getPermissionNamesResponse">
    <wsdl:part name="getPermissionNamesReturn" element="impl:GetPermissionNamesResult"/>
  </wsdl:message>

  <wsdl:message name="definePermissionRequest">
    <wsdl:part name="parms" element="impl:DefinePermission"/>
  </wsdl:message>
  <wsdl:message name="definePermissionResponse">
    <wsdl:part name="definePermissionReturn" element="impl:DefinePermissionResult"/>
  </wsdl:message>

  <wsdl:message name="updatePermissionRequest">
    <wsdl:part name="parms" element="impl:UpdatePermission"/>
  </wsdl:message>
  <wsdl:message name="updatePermissionResponse">
    <wsdl:part name="updatePermissionReturn" element="impl:UpdatePermissionResult"/>
  </wsdl:message>

  <wsdl:message name="getPermissionRequest">
    <wsdl:part name="parms" element="impl:GetPermission"/>
  </wsdl:message>
  <wsdl:message name="getPermissionResponse">
```

```
    <wsdl:part name="getPermissionReturn" element="impl:GetPermissionResult"/>
  </wsdl:message>

  <wsdl:message name="undefinePermissionRequest">
    <wsdl:part name="parms" element="impl:UndefinePermission"/>
  </wsdl:message>
  <wsdl:message name="undefinePermissionResponse">
    <wsdl:part name="undefinePermissionReturn" element="impl:UndefinePermissionResult"/>
  </wsdl:message>

  <wsdl:message name="getRoleNamesRequest">
    <wsdl:part name="parms" element="impl:GetRoleNames"/>
  </wsdl:message>
  <wsdl:message name="getRoleNamesResponse">
    <wsdl:part name="getRoleNamesReturn" element="impl:GetRoleNamesResult"/>
  </wsdl:message>

  <wsdl:message name="defineRoleRequest">
    <wsdl:part name="parms" element="impl:DefineRole"/>
  </wsdl:message>
  <wsdl:message name="defineRoleResponse">
    <wsdl:part name="defineRoleReturn" element="impl:DefineRoleResult"/>
  </wsdl:message>

  <wsdl:message name="updateRoleRequest">
    <wsdl:part name="parms" element="impl:UpdateRole"/>
  </wsdl:message>
  <wsdl:message name="updateRoleResponse">
    <wsdl:part name="updateRoleReturn" element="impl:UpdateRoleResult"/>
  </wsdl:message>

  <wsdl:message name="getRoleRequest">
    <wsdl:part name="parms" element="impl:GetRole"/>
  </wsdl:message>
  <wsdl:message name="getRoleResponse">
    <wsdl:part name="getRoleReturn" element="impl:GetRoleResult"/>
  </wsdl:message>

  <wsdl:message name="undefineRoleRequest">
    <wsdl:part name="parms" element="impl:UndefineRole"/>
  </wsdl:message>
  <wsdl:message name="undefineRoleResponse">
    <wsdl:part name="undefineRoleReturn" element="impl:UndefineRoleResult"/>
  </wsdl:message>

  <wsdl:message name="addPermissionsRequest">
    <wsdl:part name="parms" element="impl:AddPermissions"/>
  </wsdl:message>
  <wsdl:message name="addPermissionsResponse">
    <wsdl:part name="addPermissionsReturn" element="impl:AddPermissionsResult"/>
  </wsdl:message>

  <wsdl:message name="setPermissionsRequest">
    <wsdl:part name="parms" element="impl:SetPermissions"/>
  </wsdl:message>
  <wsdl:message name="setPermissionsResponse">
    <wsdl:part name="setPermissionsReturn" element="impl:SetPermissionsResult"/>
  </wsdl:message>

  <wsdl:message name="removePermissionsRequest">
    <wsdl:part name="parms" element="impl:RemovePermissions"/>
  </wsdl:message>
  <wsdl:message name="removePermissionsResponse">
    <wsdl:part name="removePermissionsReturn" element="impl:RemovePermissionsResult"/>
  </wsdl:message>

  <wsdl:message name="getClientIdentityNamesRequest">
    <wsdl:part name="parms" element="impl:GetClientIdentityNames"/>
  </wsdl:message>
  <wsdl:message name="getClientIdentityNamesResponse">
```

```
    <wsdl:part name="getClientIdentityNamesReturn"
               element="impl:GetClientIdentityNamesResult"/>
  </wsdl:message>

  <wsdl:message name="defineClientIdentityRequest">
    <wsdl:part name="parms" element="impl:DefineClientIdentity"/>
  </wsdl:message>
  <wsdl:message name="defineClientIdentityResponse">
    <wsdl:part name="defineClientIdentityReturn"
               element="impl:DefineClientIdentityResult"/>
  </wsdl:message>

  <wsdl:message name="updateClientIdentityRequest">
    <wsdl:part name="parms" element="impl:UpdateClientIdentity"/>
  </wsdl:message>
  <wsdl:message name="updateClientIdentityResponse">
    <wsdl:part name="updateClientIdentityReturn"
               element="impl:UpdateClientIdentityResult"/>
  </wsdl:message>

  <wsdl:message name="getClientIdentityRequest">
    <wsdl:part name="parms" element="impl:GetClientIdentity"/>
  </wsdl:message>
  <wsdl:message name="getClientIdentityResponse">
    <wsdl:part name="getClientIdentityReturn" element="impl:GetClientIdentityResult"/>
  </wsdl:message>

  <wsdl:message name="getClientPermissionNamesRequest">
    <wsdl:part name="parms" element="impl:GetClientPermissionNames"/>
  </wsdl:message>
  <wsdl:message name="getClientPermissionNamesResponse">
    <wsdl:part name="getClientPermissionNamesReturn"
               element="impl:GetClientPermissionNamesResult"/>
  </wsdl:message>

  <wsdl:message name="undefineClientIdentityRequest">
    <wsdl:part name="parms" element="impl:UndefineClientIdentity"/>
  </wsdl:message>
  <wsdl:message name="undefineClientIdentityResponse">
    <wsdl:part name="undefineClientIdentityReturn"
               element="impl:UndefineClientIdentityResult"/>
  </wsdl:message>

  <wsdl:message name="addRolesRequest">
    <wsdl:part name="parms" element="impl:AddRoles"/>
  </wsdl:message>
  <wsdl:message name="addRolesResponse">
    <wsdl:part name="addRolesReturn" element="impl:AddRolesResult"/>
  </wsdl:message>

  <wsdl:message name="removeRolesRequest">
    <wsdl:part name="parms" element="impl:RemoveRoles"/>
  </wsdl:message>
  <wsdl:message name="removeRolesResponse">
    <wsdl:part name="removeRolesReturn" element="impl:RemoveRolesResult"/>
  </wsdl:message>

  <wsdl:message name="setRolesRequest">
    <wsdl:part name="parms" element="impl:SetRoles"/>
  </wsdl:message>
  <wsdl:message name="setRolesResponse">
    <wsdl:part name="setRolesReturn" element="impl:SetRolesResult"/>
  </wsdl:message>

  <wsdl:message name="getSupportedOperationsRequest">
    <wsdl:part name="parms" element="impl:GetSupportedOperations"/>
  </wsdl:message>
  <wsdl:message name="getSupportedOperationsResponse">
    <wsdl:part name="getSupportedOperationsReturn"
               element="impl:GetSupportedOperationsResult"/>
  </wsdl:message>

  <wsdl:message name="getStandardVersionRequest">
    <wsdl:part name="parms" element="impl:GetStandardVersion"/>
  </wsdl:message>
  <wsdl:message name="getStandardVersionResponse">
    <wsdl:part name="getStandardVersionReturn" element="impl:GetStandardVersionResult"/>
  </wsdl:message>

  <wsdl:message name="getVendorVersionRequest">
    <wsdl:part name="parms" element="impl:GetVendorVersion"/>
  </wsdl:message>
  <wsdl:message name="getVendorVersionResponse">
    <wsdl:part name="getVendorVersionReturn" element="impl:GetVendorVersionResult"/>
  </wsdl:message>

  <wsdl:message name="SecurityExceptionResponse">
    <wsdl:part name="fault" element="impl:SecurityException"/>
  </wsdl:message>

  <wsdl:message name="NoSuchPermissionExceptionResponse">
    <wsdl:part name="fault" element="impl:NoSuchPermissionException"/>
  </wsdl:message>

  <wsdl:message name="PermissionValidationExceptionResponse">
    <wsdl:part name="fault" element="impl:PermissionValidationException"/>
  </wsdl:message>

  <wsdl:message name="DuplicatePermissionExceptionResponse">
    <wsdl:part name="fault" element="impl:DuplicatePermissionException"/>
  </wsdl:message>

  <wsdl:message name="NoSuchRoleExceptionResponse">
    <wsdl:part name="fault" element="impl:NoSuchRoleException"/>
  </wsdl:message>

  <wsdl:message name="RoleValidationExceptionResponse">
    <wsdl:part name="fault" element="impl:RoleValidationException"/>
  </wsdl:message>

  <wsdl:message name="DuplicateRoleExceptionResponse">
    <wsdl:part name="fault" element="impl:DuplicateRoleException"/>
  </wsdl:message>

  <wsdl:message name="NoSuchClientIdentityExceptionResponse">
    <wsdl:part name="fault" element="impl:NoSuchClientIdentityException"/>
  </wsdl:message>

  <wsdl:message name="ClientIdentityValidationExceptionResponse">
    <wsdl:part name="fault" element="impl:ClientIdentityValidationException"/>
  </wsdl:message>

  <wsdl:message name="DuplicateClientIdentityExceptionResponse">
    <wsdl:part name="fault" element="impl:DuplicateClientIdentityException"/>
  </wsdl:message>

  <wsdl:message name="UnsupportedOperationExceptionResponse">
    <wsdl:part name="fault" element="impl:UnsupportedOperationException"/>
  </wsdl:message>

  <wsdl:message name="ImplementationExceptionResponse">
    <wsdl:part name="fault" element="impl:ImplementationException"/>
  </wsdl:message>
  <!-- ALEACSERVICE PORTTYPE -->

  <wsdl:portType name="ALEACServicePortType">

    <wsdl:operation name="getPermissionNames">
      <wsdl:input message="impl:getPermissionNamesRequest"
                  name="getPermissionNamesRequest"/>
      <wsdl:output message="impl:getPermissionNamesResponse"
                   name="getPermissionNamesResponse"/>
      <wsdl:fault message="impl:UnsupportedOperationExceptionResponse"
                  name="UnsupportedOperationExceptionFault"/>
      <wsdl:fault message="impl:SecurityExceptionResponse"
                  name="SecurityExceptionFault"/>
      <wsdl:fault message="impl:ImplementationExceptionResponse"
                  name="ImplementationExceptionFault"/>
    </wsdl:operation>

    <wsdl:operation name="definePermission">
      <wsdl:input message="impl:definePermissionRequest" name="definePermissionRequest"/>
      <wsdl:output message="impl:definePermissionResponse"
                   name="definePermissionResponse"/>
      <wsdl:fault message="impl:SecurityExceptionResponse"
                  name="SecurityExceptionFault"/>
      <wsdl:fault message="impl:DuplicatePermissionExceptionResponse"
                  name="DuplicatePermissionExceptionFault"/>
      <wsdl:fault message="impl:PermissionValidationExceptionResponse"
                  name="PermissionValidationExceptionFault"/>
      <wsdl:fault message="impl:UnsupportedOperationExceptionResponse"
                  name="UnsupportedOperationExceptionFault"/>
      <wsdl:fault message="impl:ImplementationExceptionResponse"
                  name="ImplementationExceptionFault"/>
    </wsdl:operation>

    <wsdl:operation name="updatePermission">
      <wsdl:input message="impl:updatePermissionRequest" name="updatePermissionRequest"/>
      <wsdl:output message="impl:updatePermissionResponse"
                   name="updatePermissionResponse"/>
      <wsdl:fault message="impl:NoSuchPermissionExceptionResponse"
                  name="NoSuchPermissionExceptionFault"/>
      <wsdl:fault message="impl:PermissionValidationExceptionResponse"
                  name="PermissionValidationExceptionFault"/>
      <wsdl:fault message="impl:UnsupportedOperationExceptionResponse"
                  name="UnsupportedOperationExceptionFault"/>
      <wsdl:fault message="impl:SecurityExceptionResponse"
                  name="SecurityExceptionFault"/>
      <wsdl:fault message="impl:ImplementationExceptionResponse"
                  name="ImplementationExceptionFault"/>
    </wsdl:operation>

    <wsdl:operation name="getPermission">
      <wsdl:input message="impl:getPermissionRequest" name="getPermissionRequest"/>
      <wsdl:output message="impl:getPermissionResponse" name="getPermissionResponse"/>
      <wsdl:fault message="impl:SecurityExceptionResponse"
                  name="SecurityExceptionFault"/>
      <wsdl:fault message="impl:NoSuchPermissionExceptionResponse"
                  name="NoSuchPermissionExceptionFault"/>
      <wsdl:fault message="impl:UnsupportedOperationExceptionResponse"
                  name="UnsupportedOperationExceptionFault"/>
      <wsdl:fault message="impl:ImplementationExceptionResponse"
                  name="ImplementationExceptionFault"/>
    </wsdl:operation>

    <wsdl:operation name="undefinePermission">
      <wsdl:input message="impl:undefinePermissionRequest"
                  name="undefinePermissionRequest"/>
      <wsdl:output message="impl:undefinePermissionResponse"
                   name="undefinePermissionResponse"/>
      <wsdl:fault message="impl:SecurityExceptionResponse"
                  name="SecurityExceptionFault"/>
      <wsdl:fault message="impl:NoSuchPermissionExceptionResponse"
                  name="NoSuchPermissionExceptionFault"/>
      <wsdl:fault message="impl:UnsupportedOperationExceptionResponse"
                  name="UnsupportedOperationExceptionFault"/>
      <wsdl:fault message="impl:ImplementationExceptionResponse"
                  name="ImplementationExceptionFault"/>
    </wsdl:operation>
```

```
      <wsdl:operation name="getRoleNames">
        <wsdl:input message="impl:getRoleNamesRequest" name="getRoleNamesRequest"/>
        <wsdl:output message="impl:getRoleNamesResponse" name="getRoleNamesResponse"/>
        <wsdl:fault message="impl:SecurityExceptionResponse"
                    name="SecurityExceptionFault"/>
        <wsdl:fault message="impl:UnsupportedOperationExceptionResponse"
                    name="UnsupportedOperationExceptionFault"/>
        <wsdl:fault message="impl:ImplementationExceptionResponse"
                    name="ImplementationExceptionFault"/>
      </wsdl:operation>

      <wsdl:operation name="defineRole">
        <wsdl:input message="impl:defineRoleRequest" name="defineRoleRequest"/>
        <wsdl:output message="impl:defineRoleResponse" name="defineRoleResponse"/>
        <wsdl:fault message="impl:SecurityExceptionResponse"
                    name="SecurityExceptionFault"/>
        <wsdl:fault message="impl:DuplicateRoleExceptionResponse"
                    name="DuplicateRoleExceptionFault"/>
        <wsdl:fault message="impl:RoleValidationExceptionResponse"
                    name="RoleValidationExceptionFault"/>
        <wsdl:fault message="impl:UnsupportedOperationExceptionResponse"
                    name="UnsupportedOperationExceptionFault"/>
        <wsdl:fault message="impl:ImplementationExceptionResponse"
                    name="ImplementationExceptionFault"/>
      </wsdl:operation>

      <wsdl:operation name="updateRole">
        <wsdl:input message="impl:updateRoleRequest" name="updateRoleRequest"/>
        <wsdl:output message="impl:updateRoleResponse" name="updateRoleResponse"/>
        <wsdl:fault message="impl:NoSuchRoleExceptionResponse"
                    name="NoSuchRoleExceptionFault"/>
        <wsdl:fault message="impl:RoleValidationExceptionResponse"
                    name="RoleValidationExceptionFault"/>
        <wsdl:fault message="impl:UnsupportedOperationExceptionResponse"
                    name="UnsupportedOperationExceptionFault"/>
        <wsdl:fault message="impl:SecurityExceptionResponse"
                    name="SecurityExceptionFault"/>
        <wsdl:fault message="impl:ImplementationExceptionResponse"
                    name="ImplementationExceptionFault"/>
      </wsdl:operation>

      <wsdl:operation name="getRole">
        <wsdl:input message="impl:getRoleRequest" name="getRoleRequest"/>
        <wsdl:output message="impl:getRoleResponse" name="getRoleResponse"/>
        <wsdl:fault message="impl:SecurityExceptionResponse"
                    name="SecurityExceptionFault"/>
        <wsdl:fault message="impl:NoSuchRoleExceptionResponse"
                    name="NoSuchRoleExceptionFault"/>
        <wsdl:fault message="impl:UnsupportedOperationExceptionResponse"
                    name="UnsupportedOperationExceptionFault"/>
        <wsdl:fault message="impl:ImplementationExceptionResponse"
                    name="ImplementationExceptionFault"/>
      </wsdl:operation>

      <wsdl:operation name="undefineRole">
        <wsdl:input message="impl:undefineRoleRequest" name="undefineRoleRequest"/>
        <wsdl:output message="impl:undefineRoleResponse" name="undefineRoleResponse"/>
        <wsdl:fault message="impl:SecurityExceptionResponse"
                    name="SecurityExceptionFault"/>
        <wsdl:fault message="impl:NoSuchRoleExceptionResponse"
                    name="NoSuchRoleExceptionFault"/>
        <wsdl:fault message="impl:UnsupportedOperationExceptionResponse"
                    name="UnsupportedOperationExceptionFault"/>
        <wsdl:fault message="impl:ImplementationExceptionResponse"
                    name="ImplementationExceptionFault"/>
      </wsdl:operation>

      <wsdl:operation name="addPermissions">
        <wsdl:input message="impl:addPermissionsRequest" name="addPermissionsRequest"/>
        <wsdl:output message="impl:addPermissionsResponse" name="addPermissionsResponse"/>
        <wsdl:fault message="impl:SecurityExceptionResponse"
                    name="SecurityExceptionFault"/>
        <wsdl:fault message="impl:NoSuchRoleExceptionResponse"
                    name="NoSuchRoleExceptionFault"/>
        <wsdl:fault message="impl:NoSuchPermissionExceptionResponse"
                    name="NoSuchPermissionExceptionFault"/>
        <wsdl:fault message="impl:UnsupportedOperationExceptionResponse"
                    name="UnsupportedOperationExceptionFault"/>
        <wsdl:fault message="impl:ImplementationExceptionResponse"
                    name="ImplementationExceptionFault"/>
      </wsdl:operation>

      <wsdl:operation name="setPermissions">
        <wsdl:input message="impl:setPermissionsRequest" name="setPermissionsRequest"/>
        <wsdl:output message="impl:setPermissionsResponse" name="setPermissionsResponse"/>
        <wsdl:fault message="impl:SecurityExceptionResponse"
                    name="SecurityExceptionFault"/>
        <wsdl:fault message="impl:NoSuchRoleExceptionResponse"
                    name="NoSuchRoleExceptionFault"/>
        <wsdl:fault message="impl:NoSuchPermissionExceptionResponse"
                    name="NoSuchPermissionExceptionFault"/>
        <wsdl:fault message="impl:UnsupportedOperationExceptionResponse"
                    name="UnsupportedOperationExceptionFault"/>
        <wsdl:fault message="impl:ImplementationExceptionResponse"
                    name="ImplementationExceptionFault"/>
      </wsdl:operation>

      <wsdl:operation name="removePermissions">
        <wsdl:input message="impl:removePermissionsRequest"
                    name="removePermissionsRequest"/>
        <wsdl:output message="impl:removePermissionsResponse"
                     name="removePermissionsResponse"/>
        <wsdl:fault message="impl:SecurityExceptionResponse"
                    name="SecurityExceptionFault"/>
        <wsdl:fault message="impl:NoSuchRoleExceptionResponse"
                    name="NoSuchRoleExceptionFault"/>
        <wsdl:fault message="impl:UnsupportedOperationExceptionResponse"
                    name="UnsupportedOperationExceptionFault"/>
        <wsdl:fault message="impl:ImplementationExceptionResponse"
                    name="ImplementationExceptionFault"/>
      </wsdl:operation>

      <wsdl:operation name="getClientIdentityNames">
        <wsdl:input message="impl:getClientIdentityNamesRequest"
                    name="getClientIdentityNamesRequest"/>
        <wsdl:output message="impl:getClientIdentityNamesResponse"
                     name="getClientIdentityNamesResponse"/>
        <wsdl:fault message="impl:UnsupportedOperationExceptionResponse"
                    name="UnsupportedOperationExceptionFault"/>
        <wsdl:fault message="impl:SecurityExceptionResponse"
                    name="SecurityExceptionFault"/>
        <wsdl:fault message="impl:ImplementationExceptionResponse"
                    name="ImplementationExceptionFault"/>
      </wsdl:operation>

      <wsdl:operation name="defineClientIdentity">
        <wsdl:input message="impl:defineClientIdentityRequest"
                    name="defineClientIdentityRequest"/>
        <wsdl:output message="impl:defineClientIdentityResponse"
                     name="defineClientIdentityResponse"/>
        <wsdl:fault message="impl:SecurityExceptionResponse"
                    name="SecurityExceptionFault"/>
        <wsdl:fault message="impl:DuplicateClientIdentityExceptionResponse"
                    name="DuplicateClientIdentityExceptionFault"/>
        <wsdl:fault message="impl:ClientIdentityValidationExceptionResponse"
                    name="ClientIdentityValidationExceptionFault"/>
        <wsdl:fault message="impl:UnsupportedOperationExceptionResponse"
                    name="UnsupportedOperationExceptionFault"/>
        <wsdl:fault message="impl:ImplementationExceptionResponse"
                    name="ImplementationExceptionFault"/>
```

```
        </wsdl:operation>

        <wsdl:operation name="updateClientIdentity">
          <wsdl:input message="impl:updateClientIdentityRequest"
                      name="updateClientIdentityRequest"/>
          <wsdl:output message="impl:updateClientIdentityResponse"
                       name="updateClientIdentityResponse"/>
          <wsdl:fault message="impl:SecurityExceptionResponse"
                      name="SecurityExceptionFault"/>
          <wsdl:fault message="impl:NoSuchClientIdentityExceptionResponse"
                      name="NoSuchClientIdentityExceptionFault"/>
          <wsdl:fault message="impl:ClientIdentityValidationExceptionResponse"
                      name="ClientIdentityValidationExceptionFault"/>
          <wsdl:fault message="impl:UnsupportedOperationExceptionResponse"
                      name="UnsupportedOperationExceptionFault"/>
          <wsdl:fault message="impl:ImplementationExceptionResponse"
                      name="ImplementationExceptionFault"/>
        </wsdl:operation>

        <wsdl:operation name="getClientIdentity">
          <wsdl:input message="impl:getClientIdentityRequest"
                      name="getClientIdentityRequest"/>
          <wsdl:output message="impl:getClientIdentityResponse"
                       name="getClientIdentityResponse"/>
          <wsdl:fault message="impl:SecurityExceptionResponse"
                      name="SecurityExceptionFault"/>
          <wsdl:fault message="impl:NoSuchClientIdentityExceptionResponse"
                      name="NoSuchClientIdentityExceptionFault"/>
          <wsdl:fault message="impl:UnsupportedOperationExceptionResponse"
                      name="UnsupportedOperationExceptionFault"/>
          <wsdl:fault message="impl:ImplementationExceptionResponse"
                      name="ImplementationExceptionFault"/>
        </wsdl:operation>

        <wsdl:operation name="getClientPermissionNames">
          <wsdl:input message="impl:getClientPermissionNamesRequest"
                      name="getClientPermissionNamesRequest"/>
          <wsdl:output message="impl:getClientPermissionNamesResponse"
                       name="getClientPermissionNamesResponse"/>
          <wsdl:fault message="impl:SecurityExceptionResponse"
                      name="SecurityExceptionFault"/>
          <wsdl:fault message="impl:NoSuchClientIdentityExceptionResponse"
                      name="NoSuchClientIdentityExceptionFault"/>
          <wsdl:fault message="impl:UnsupportedOperationExceptionResponse"
                      name="UnsupportedOperationExceptionFault"/>
          <wsdl:fault message="impl:ImplementationExceptionResponse"
                      name="ImplementationExceptionFault"/>
        </wsdl:operation>

        <wsdl:operation name="undefineClientIdentity">
          <wsdl:input message="impl:undefineClientIdentityRequest"
                      name="undefineClientIdentityRequest"/>
          <wsdl:output message="impl:undefineClientIdentityResponse"
                       name="undefineClientIdentityResponse"/>
          <wsdl:fault message="impl:SecurityExceptionResponse"
                      name="SecurityExceptionFault"/>
          <wsdl:fault message="impl:NoSuchClientIdentityExceptionResponse"
                      name="NoSuchClientIdentityExceptionFault"/>
          <wsdl:fault message="impl:UnsupportedOperationExceptionResponse"
                      name="UnsupportedOperationExceptionFault"/>
          <wsdl:fault message="impl:ImplementationExceptionResponse"
                      name="ImplementationExceptionFault"/>
        </wsdl:operation>

        <wsdl:operation name="addRoles">
          <wsdl:input message="impl:addRolesRequest" name="addRolesRequest"/>
          <wsdl:output message="impl:addRolesResponse" name="addRolesResponse"/>
          <wsdl:fault message="impl:SecurityExceptionResponse"
                      name="SecurityExceptionFault"/>
          <wsdl:fault message="impl:NoSuchClientIdentityExceptionResponse"
```

```
                  name="NoSuchClientIdentityExceptionFault"/>
      <wsdl:fault message="impl:NoSuchRoleExceptionResponse"
                  name="NoSuchRoleExceptionFault"/>
      <wsdl:fault message="impl:UnsupportedOperationExceptionResponse"
                  name="UnsupportedOperationExceptionFault"/>
      <wsdl:fault message="impl:ImplementationExceptionResponse"
                  name="ImplementationExceptionFault"/>
    </wsdl:operation>

    <wsdl:operation name="removeRoles">
      <wsdl:input message="impl:removeRolesRequest" name="removeRolesRequest"/>
      <wsdl:output message="impl:removeRolesResponse" name="removeRolesResponse"/>
      <wsdl:fault message="impl:SecurityExceptionResponse"
                  name="SecurityExceptionFault"/>
      <wsdl:fault message="impl:NoSuchClientIdentityExceptionResponse"
                  name="NoSuchClientIdentityExceptionFault"/>
      <wsdl:fault message="impl:UnsupportedOperationExceptionResponse"
                  name="UnsupportedOperationExceptionFault"/>
      <wsdl:fault message="impl:ImplementationExceptionResponse"
                  name="ImplementationExceptionFault"/>
    </wsdl:operation>

    <wsdl:operation name="setRoles">
      <wsdl:input message="impl:setRolesRequest" name="setRolesRequest"/>
      <wsdl:output message="impl:setRolesResponse" name="setRolesResponse"/>
      <wsdl:fault message="impl:SecurityExceptionResponse"
                  name="SecurityExceptionFault"/>
      <wsdl:fault message="impl:NoSuchClientIdentityExceptionResponse"
                  name="NoSuchClientIdentityExceptionFault"/>
      <wsdl:fault message="impl:NoSuchRoleExceptionResponse"
                  name="NoSuchRoleExceptionFault"/>
      <wsdl:fault message="impl:UnsupportedOperationExceptionResponse"
                  name="UnsupportedOperationExceptionFault"/>
      <wsdl:fault message="impl:ImplementationExceptionResponse"
                  name="ImplementationExceptionFault"/>
    </wsdl:operation>

    <wsdl:operation name="getSupportedOperations">
      <wsdl:input message="impl:getSupportedOperationsRequest"
                  name="getSupportedOperationsRequest"/>
      <wsdl:output message="impl:getSupportedOperationsResponse"
                   name="getSupportedOperationsResponse"/>
      <wsdl:fault message="impl:ImplementationExceptionResponse"
                  name="ImplementationExceptionFault"/>
    </wsdl:operation>

    <wsdl:operation name="getStandardVersion">
      <wsdl:input message="impl:getStandardVersionRequest"
                  name="getStandardVersionRequest"/>
      <wsdl:output message="impl:getStandardVersionResponse"
                   name="getStandardVersionResponse"/>
      <wsdl:fault message="impl:ImplementationExceptionResponse"
                  name="ImplementationExceptionFault"/>
    </wsdl:operation>

    <wsdl:operation name="getVendorVersion">
      <wsdl:input message="impl:getVendorVersionRequest" name="getVendorVersionRequest"/>
      <wsdl:output message="impl:getVendorVersionResponse"
                   name="getVendorVersionResponse"/>
      <wsdl:fault message="impl:ImplementationExceptionResponse"
                  name="ImplementationExceptionFault"/>
    </wsdl:operation>
  </wsdl:portType>
  <!-- ALEACSERVICE BINDING -->

  <wsdl:binding name="ALEACServiceBinding" type="impl:ALEACServicePortType">
    <wsdlsoap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>

    <wsdl:operation name="getPermissionNames">
      <wsdlsoap:operation soapAction=""/>
```

```
        <wsdl:input name="getPermissionNamesRequest">
          <wsdlsoap:body use="literal"/>
        </wsdl:input>
        <wsdl:output name="getPermissionNamesResponse">
          <wsdlsoap:body use="literal"/>
        </wsdl:output>
        <wsdl:fault name="UnsupportedOperationExceptionFault">
          <wsdlsoap:fault name="UnsupportedOperationExceptionFault" use="literal"/>
        </wsdl:fault>
        <wsdl:fault name="SecurityExceptionFault">
          <wsdlsoap:fault name="SecurityExceptionFault" use="literal"/>
        </wsdl:fault>
        <wsdl:fault name="ImplementationExceptionFault">
          <wsdlsoap:fault name="ImplementationExceptionFault" use="literal"/>
        </wsdl:fault>
      </wsdl:operation>

      <wsdl:operation name="definePermission">
        <wsdlsoap:operation soapAction=""/>
        <wsdl:input name="definePermissionRequest">
          <wsdlsoap:body use="literal"/>
        </wsdl:input>
        <wsdl:output name="definePermissionResponse">
          <wsdlsoap:body use="literal"/>
        </wsdl:output>
        <wsdl:fault name="SecurityExceptionFault">
          <wsdlsoap:fault name="SecurityExceptionFault" use="literal"/>
        </wsdl:fault>
        <wsdl:fault name="DuplicatePermissionExceptionFault">
          <wsdlsoap:fault name="DuplicatePermissionExceptionFault" use="literal"/>
        </wsdl:fault>
        <wsdl:fault name="PermissionValidationExceptionFault">
          <wsdlsoap:fault name="PermissionValidationExceptionFault" use="literal"/>
        </wsdl:fault>
        <wsdl:fault name="UnsupportedOperationExceptionFault">
          <wsdlsoap:fault name="UnsupportedOperationExceptionFault" use="literal"/>
        </wsdl:fault>
        <wsdl:fault name="ImplementationExceptionFault">
          <wsdlsoap:fault name="ImplementationExceptionFault" use="literal"/>
        </wsdl:fault>
      </wsdl:operation>

      <wsdl:operation name="updatePermission">
        <wsdlsoap:operation soapAction=""/>
        <wsdl:input name="updatePermissionRequest">
          <wsdlsoap:body use="literal"/>
        </wsdl:input>
        <wsdl:output name="updatePermissionResponse">
          <wsdlsoap:body use="literal"/>
        </wsdl:output>
        <wsdl:fault name="NoSuchPermissionExceptionFault">
          <wsdlsoap:fault name="NoSuchPermissionExceptionFault" use="literal"/>
        </wsdl:fault>
        <wsdl:fault name="PermissionValidationExceptionFault">
          <wsdlsoap:fault name="PermissionValidationExceptionFault" use="literal"/>
        </wsdl:fault>
        <wsdl:fault name="UnsupportedOperationExceptionFault">
          <wsdlsoap:fault name="UnsupportedOperationExceptionFault" use="literal"/>
        </wsdl:fault>
        <wsdl:fault name="SecurityExceptionFault">
          <wsdlsoap:fault name="SecurityExceptionFault" use="literal"/>
        </wsdl:fault>
        <wsdl:fault name="ImplementationExceptionFault">
          <wsdlsoap:fault name="ImplementationExceptionFault" use="literal"/>
        </wsdl:fault>
      </wsdl:operation>

      <wsdl:operation name="getPermission">
        <wsdlsoap:operation soapAction=""/>
        <wsdl:input name="getPermissionRequest">
```

```
          <wsdlsoap:body use="literal"/>
        </wsdl:input>
        <wsdl:output name="getPermissionResponse">
          <wsdlsoap:body use="literal"/>
        </wsdl:output>
        <wsdl:fault name="SecurityExceptionFault">
          <wsdlsoap:fault name="SecurityExceptionFault" use="literal"/>
        </wsdl:fault>
        <wsdl:fault name="NoSuchPermissionExceptionFault">
          <wsdlsoap:fault name="NoSuchPermissionExceptionFault" use="literal"/>
        </wsdl:fault>
        <wsdl:fault name="UnsupportedOperationExceptionFault">
          <wsdlsoap:fault name="UnsupportedOperationExceptionFault" use="literal"/>
        </wsdl:fault>
        <wsdl:fault name="ImplementationExceptionFault">
          <wsdlsoap:fault name="ImplementationExceptionFault" use="literal"/>
        </wsdl:fault>
      </wsdl:operation>

      <wsdl:operation name="undefinePermission">
        <wsdlsoap:operation soapAction=""/>
        <wsdl:input name="undefinePermissionRequest">
          <wsdlsoap:body use="literal"/>
        </wsdl:input>
        <wsdl:output name="undefinePermissionResponse">
          <wsdlsoap:body use="literal"/>
        </wsdl:output>
        <wsdl:fault name="SecurityExceptionFault">
          <wsdlsoap:fault name="SecurityExceptionFault" use="literal"/>
        </wsdl:fault>
        <wsdl:fault name="NoSuchPermissionExceptionFault">
          <wsdlsoap:fault name="NoSuchPermissionExceptionFault" use="literal"/>
        </wsdl:fault>
        <wsdl:fault name="UnsupportedOperationExceptionFault">
          <wsdlsoap:fault name="UnsupportedOperationExceptionFault" use="literal"/>
        </wsdl:fault>
        <wsdl:fault name="ImplementationExceptionFault">
          <wsdlsoap:fault name="ImplementationExceptionFault" use="literal"/>
        </wsdl:fault>
      </wsdl:operation>

      <wsdl:operation name="getRoleNames">
        <wsdlsoap:operation soapAction=""/>
        <wsdl:input name="getRoleNamesRequest">
          <wsdlsoap:body use="literal"/>
        </wsdl:input>
        <wsdl:output name="getRoleNamesResponse">
          <wsdlsoap:body use="literal"/>
        </wsdl:output>
        <wsdl:fault name="SecurityExceptionFault">
          <wsdlsoap:fault name="SecurityExceptionFault" use="literal"/>
        </wsdl:fault>
        <wsdl:fault name="UnsupportedOperationExceptionFault">
          <wsdlsoap:fault name="UnsupportedOperationExceptionFault" use="literal"/>
        </wsdl:fault>
        <wsdl:fault name="ImplementationExceptionFault">
          <wsdlsoap:fault name="ImplementationExceptionFault" use="literal"/>
        </wsdl:fault>
      </wsdl:operation>

      <wsdl:operation name="defineRole">
        <wsdlsoap:operation soapAction=""/>
        <wsdl:input name="defineRoleRequest">
          <wsdlsoap:body use="literal"/>
        </wsdl:input>
        <wsdl:output name="defineRoleResponse">
          <wsdlsoap:body use="literal"/>
        </wsdl:output>
        <wsdl:fault name="SecurityExceptionFault">
          <wsdlsoap:fault name="SecurityExceptionFault" use="literal"/>
        </wsdl:fault>
        <wsdl:fault name="DuplicateRoleExceptionFault">
          <wsdlsoap:fault name="DuplicateRoleExceptionFault" use="literal"/>
        </wsdl:fault>
        <wsdl:fault name="RoleValidationExceptionFault">
          <wsdlsoap:fault name="RoleValidationExceptionFault" use="literal"/>
        </wsdl:fault>
        <wsdl:fault name="UnsupportedOperationExceptionFault">
          <wsdlsoap:fault name="UnsupportedOperationExceptionFault" use="literal"/>
        </wsdl:fault>
        <wsdl:fault name="ImplementationExceptionFault">
          <wsdlsoap:fault name="ImplementationExceptionFault" use="literal"/>
        </wsdl:fault>
      </wsdl:operation>

      <wsdl:operation name="updateRole">
        <wsdlsoap:operation soapAction=""/>
        <wsdl:input name="updateRoleRequest">
          <wsdlsoap:body use="literal"/>
        </wsdl:input>
        <wsdl:output name="updateRoleResponse">
          <wsdlsoap:body use="literal"/>
        </wsdl:output>
        <wsdl:fault name="NoSuchRoleExceptionFault">
          <wsdlsoap:fault name="NoSuchRoleExceptionFault" use="literal"/>
        </wsdl:fault>
        <wsdl:fault name="RoleValidationExceptionFault">
          <wsdlsoap:fault name="RoleValidationExceptionFault" use="literal"/>
        </wsdl:fault>
        <wsdl:fault name="UnsupportedOperationExceptionFault">
          <wsdlsoap:fault name="UnsupportedOperationExceptionFault" use="literal"/>
        </wsdl:fault>
        <wsdl:fault name="SecurityExceptionFault">
          <wsdlsoap:fault name="SecurityExceptionFault" use="literal"/>
        </wsdl:fault>
        <wsdl:fault name="ImplementationExceptionFault">
          <wsdlsoap:fault name="ImplementationExceptionFault" use="literal"/>
        </wsdl:fault>
      </wsdl:operation>

      <wsdl:operation name="getRole">
        <wsdlsoap:operation soapAction=""/>
        <wsdl:input name="getRoleRequest">
          <wsdlsoap:body use="literal"/>
        </wsdl:input>
        <wsdl:output name="getRoleResponse">
          <wsdlsoap:body use="literal"/>
        </wsdl:output>
        <wsdl:fault name="SecurityExceptionFault">
          <wsdlsoap:fault name="SecurityExceptionFault" use="literal"/>
        </wsdl:fault>
        <wsdl:fault name="NoSuchRoleExceptionFault">
          <wsdlsoap:fault name="NoSuchRoleExceptionFault" use="literal"/>
        </wsdl:fault>
        <wsdl:fault name="UnsupportedOperationExceptionFault">
          <wsdlsoap:fault name="UnsupportedOperationExceptionFault" use="literal"/>
        </wsdl:fault>
        <wsdl:fault name="ImplementationExceptionFault">
          <wsdlsoap:fault name="ImplementationExceptionFault" use="literal"/>
        </wsdl:fault>
      </wsdl:operation>

      <wsdl:operation name="undefineRole">
        <wsdlsoap:operation soapAction=""/>
        <wsdl:input name="undefineRoleRequest">
          <wsdlsoap:body use="literal"/>
        </wsdl:input>
        <wsdl:output name="undefineRoleResponse">
          <wsdlsoap:body use="literal"/>
        </wsdl:output>
        <wsdl:fault name="SecurityExceptionFault">
          <wsdlsoap:fault name="SecurityExceptionFault" use="literal"/>
        </wsdl:fault>
        <wsdl:fault name="NoSuchRoleExceptionFault">
          <wsdlsoap:fault name="NoSuchRoleExceptionFault" use="literal"/>
        </wsdl:fault>
        <wsdl:fault name="UnsupportedOperationExceptionFault">
          <wsdlsoap:fault name="UnsupportedOperationExceptionFault" use="literal"/>
        </wsdl:fault>
        <wsdl:fault name="ImplementationExceptionFault">
          <wsdlsoap:fault name="ImplementationExceptionFault" use="literal"/>
        </wsdl:fault>
      </wsdl:operation>

      <wsdl:operation name="addPermissions">
        <wsdlsoap:operation soapAction=""/>
        <wsdl:input name="addPermissionsRequest">
          <wsdlsoap:body use="literal"/>
        </wsdl:input>
        <wsdl:output name="addPermissionsResponse">
          <wsdlsoap:body use="literal"/>
        </wsdl:output>
        <wsdl:fault name="SecurityExceptionFault">
          <wsdlsoap:fault name="SecurityExceptionFault" use="literal"/>
        </wsdl:fault>
        <wsdl:fault name="NoSuchRoleExceptionFault">
          <wsdlsoap:fault name="NoSuchRoleExceptionFault" use="literal"/>
        </wsdl:fault>
        <wsdl:fault name="NoSuchPermissionExceptionFault">
          <wsdlsoap:fault name="NoSuchPermissionExceptionFault" use="literal"/>
        </wsdl:fault>
        <wsdl:fault name="UnsupportedOperationExceptionFault">
          <wsdlsoap:fault name="UnsupportedOperationExceptionFault" use="literal"/>
        </wsdl:fault>
        <wsdl:fault name="ImplementationExceptionFault">
          <wsdlsoap:fault name="ImplementationExceptionFault" use="literal"/>
        </wsdl:fault>
      </wsdl:operation>

      <wsdl:operation name="setPermissions">
        <wsdlsoap:operation soapAction=""/>
        <wsdl:input name="setPermissionsRequest">
          <wsdlsoap:body use="literal"/>
        </wsdl:input>
        <wsdl:output name="setPermissionsResponse">
          <wsdlsoap:body use="literal"/>
        </wsdl:output>
        <wsdl:fault name="SecurityExceptionFault">
          <wsdlsoap:fault name="SecurityExceptionFault" use="literal"/>
        </wsdl:fault>
        <wsdl:fault name="NoSuchRoleExceptionFault">
          <wsdlsoap:fault name="NoSuchRoleExceptionFault" use="literal"/>
        </wsdl:fault>
        <wsdl:fault name="NoSuchPermissionExceptionFault">
          <wsdlsoap:fault name="NoSuchPermissionExceptionFault" use="literal"/>
        </wsdl:fault>
        <wsdl:fault name="UnsupportedOperationExceptionFault">
          <wsdlsoap:fault name="UnsupportedOperationExceptionFault" use="literal"/>
        </wsdl:fault>
        <wsdl:fault name="ImplementationExceptionFault">
          <wsdlsoap:fault name="ImplementationExceptionFault" use="literal"/>
        </wsdl:fault>
      </wsdl:operation>

      <wsdl:operation name="removePermissions">
        <wsdlsoap:operation soapAction=""/>
        <wsdl:input name="removePermissionsRequest">
          <wsdlsoap:body use="literal"/>
        </wsdl:input>
        <wsdl:output name="removePermissionsResponse">
```

```
        <wsdlsoap:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="SecurityExceptionFault">
        <wsdlsoap:fault name="SecurityExceptionFault" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="NoSuchRoleExceptionFault">
        <wsdlsoap:fault name="NoSuchRoleExceptionFault" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="UnsupportedOperationExceptionFault">
        <wsdlsoap:fault name="UnsupportedOperationExceptionFault" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="ImplementationExceptionFault">
        <wsdlsoap:fault name="ImplementationExceptionFault" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>

    <wsdl:operation name="getClientIdentityNames">
      <wsdlsoap:operation soapAction=""/>
      <wsdl:input name="getClientIdentityNamesRequest">
        <wsdlsoap:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="getClientIdentityNamesResponse">
        <wsdlsoap:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="UnsupportedOperationExceptionFault">
        <wsdlsoap:fault name="UnsupportedOperationExceptionFault" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="SecurityExceptionFault">
        <wsdlsoap:fault name="SecurityExceptionFault" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="ImplementationExceptionFault">
        <wsdlsoap:fault name="ImplementationExceptionFault" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>

    <wsdl:operation name="defineClientIdentity">
      <wsdlsoap:operation soapAction=""/>
      <wsdl:input name="defineClientIdentityRequest">
        <wsdlsoap:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="defineClientIdentityResponse">
        <wsdlsoap:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="SecurityExceptionFault">
        <wsdlsoap:fault name="SecurityExceptionFault" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="DuplicateClientIdentityExceptionFault">
        <wsdlsoap:fault name="DuplicateClientIdentityExceptionFault" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="ClientIdentityValidationExceptionFault">
        <wsdlsoap:fault name="ClientIdentityValidationExceptionFault" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="UnsupportedOperationExceptionFault">
        <wsdlsoap:fault name="UnsupportedOperationExceptionFault" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="ImplementationExceptionFault">
        <wsdlsoap:fault name="ImplementationExceptionFault" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>

    <wsdl:operation name="updateClientIdentity">
      <wsdlsoap:operation soapAction=""/>
      <wsdl:input name="updateClientIdentityRequest">
        <wsdlsoap:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="updateClientIdentityResponse">
        <wsdlsoap:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="SecurityExceptionFault">
        <wsdlsoap:fault name="SecurityExceptionFault" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="NoSuchClientIdentityExceptionFault">
        <wsdlsoap:fault name="NoSuchClientIdentityExceptionFault" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="ClientIdentityValidationExceptionFault">
        <wsdlsoap:fault name="ClientIdentityValidationExceptionFault" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="UnsupportedOperationExceptionFault">
        <wsdlsoap:fault name="UnsupportedOperationExceptionFault" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="ImplementationExceptionFault">
        <wsdlsoap:fault name="ImplementationExceptionFault" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>

    <wsdl:operation name="getClientIdentity">
      <wsdlsoap:operation soapAction=""/>
      <wsdl:input name="getClientIdentityRequest">
        <wsdlsoap:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="getClientIdentityResponse">
        <wsdlsoap:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="SecurityExceptionFault">
        <wsdlsoap:fault name="SecurityExceptionFault" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="NoSuchClientIdentityExceptionFault">
        <wsdlsoap:fault name="NoSuchClientIdentityExceptionFault" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="UnsupportedOperationExceptionFault">
        <wsdlsoap:fault name="UnsupportedOperationExceptionFault" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="ImplementationExceptionFault">
        <wsdlsoap:fault name="ImplementationExceptionFault" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>

    <wsdl:operation name="getClientPermissionNames">
      <wsdlsoap:operation soapAction=""/>
      <wsdl:input name="getClientPermissionNamesRequest">
        <wsdlsoap:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="getClientPermissionNamesResponse">
        <wsdlsoap:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="SecurityExceptionFault">
        <wsdlsoap:fault name="SecurityExceptionFault" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="NoSuchClientIdentityExceptionFault">
        <wsdlsoap:fault name="NoSuchClientIdentityExceptionFault" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="UnsupportedOperationExceptionFault">
        <wsdlsoap:fault name="UnsupportedOperationExceptionFault" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="ImplementationExceptionFault">
        <wsdlsoap:fault name="ImplementationExceptionFault" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>

    <wsdl:operation name="undefineClientIdentity">
      <wsdlsoap:operation soapAction=""/>
      <wsdl:input name="undefineClientIdentityRequest">
        <wsdlsoap:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="undefineClientIdentityResponse">
        <wsdlsoap:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="SecurityExceptionFault">
        <wsdlsoap:fault name="SecurityExceptionFault" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="NoSuchClientIdentityExceptionFault">
        <wsdlsoap:fault name="NoSuchClientIdentityExceptionFault" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="UnsupportedOperationExceptionFault">
        <wsdlsoap:fault name="UnsupportedOperationExceptionFault" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="ImplementationExceptionFault">
        <wsdlsoap:fault name="ImplementationExceptionFault" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>

    <wsdl:operation name="addRoles">
      <wsdlsoap:operation soapAction=""/>
      <wsdl:input name="addRolesRequest">
        <wsdlsoap:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="addRolesResponse">
        <wsdlsoap:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="SecurityExceptionFault">
        <wsdlsoap:fault name="SecurityExceptionFault" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="NoSuchClientIdentityExceptionFault">
        <wsdlsoap:fault name="NoSuchClientIdentityExceptionFault" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="NoSuchRoleExceptionFault">
        <wsdlsoap:fault name="NoSuchRoleExceptionFault" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="UnsupportedOperationExceptionFault">
        <wsdlsoap:fault name="UnsupportedOperationExceptionFault" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="ImplementationExceptionFault">
        <wsdlsoap:fault name="ImplementationExceptionFault" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>

    <wsdl:operation name="removeRoles">
      <wsdlsoap:operation soapAction=""/>
      <wsdl:input name="removeRolesRequest">
        <wsdlsoap:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="removeRolesResponse">
        <wsdlsoap:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="SecurityExceptionFault">
        <wsdlsoap:fault name="SecurityExceptionFault" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="NoSuchClientIdentityExceptionFault">
        <wsdlsoap:fault name="NoSuchClientIdentityExceptionFault" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="UnsupportedOperationExceptionFault">
        <wsdlsoap:fault name="UnsupportedOperationExceptionFault" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="ImplementationExceptionFault">
        <wsdlsoap:fault name="ImplementationExceptionFault" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>

    <wsdl:operation name="setRoles">
      <wsdlsoap:operation soapAction=""/>
      <wsdl:input name="setRolesRequest">
        <wsdlsoap:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="setRolesResponse">
        <wsdlsoap:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="SecurityExceptionFault">
        <wsdlsoap:fault name="SecurityExceptionFault" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="NoSuchClientIdentityExceptionFault">
```

```
          <wsdlsoap:fault name="NoSuchClientIdentityExceptionFault" use="literal"/>
        </wsdl:fault>
        <wsdl:fault name="NoSuchRoleExceptionFault">
          <wsdlsoap:fault name="NoSuchRoleExceptionFault" use="literal"/>
        </wsdl:fault>
        <wsdl:fault name="UnsupportedOperationExceptionFault">
          <wsdlsoap:fault name="UnsupportedOperationExceptionFault" use="literal"/>
        </wsdl:fault>
        <wsdl:fault name="ImplementationExceptionFault">
          <wsdlsoap:fault name="ImplementationExceptionFault" use="literal"/>
        </wsdl:fault>
      </wsdl:operation>

      <wsdl:operation name="getSupportedOperations">
        <wsdlsoap:operation soapAction=""/>
        <wsdl:input name="getSupportedOperationsRequest">
          <wsdlsoap:body use="literal"/>
        </wsdl:input>
        <wsdl:output name="getSupportedOperationsResponse">
          <wsdlsoap:body use="literal"/>
        </wsdl:output>
        <wsdl:fault name="ImplementationExceptionFault">
          <wsdlsoap:fault name="ImplementationExceptionFault" use="literal"/>
        </wsdl:fault>
      </wsdl:operation>

      <wsdl:operation name="getStandardVersion">
        <wsdlsoap:operation soapAction=""/>
        <wsdl:input name="getStandardVersionRequest">
          <wsdlsoap:body use="literal"/>
        </wsdl:input>
        <wsdl:output name="getStandardVersionResponse">
          <wsdlsoap:body use="literal"/>
        </wsdl:output>
        <wsdl:fault name="ImplementationExceptionFault">
          <wsdlsoap:fault name="ImplementationExceptionFault" use="literal"/>
        </wsdl:fault>
      </wsdl:operation>

      <wsdl:operation name="getVendorVersion">
        <wsdlsoap:operation soapAction=""/>
        <wsdl:input name="getVendorVersionRequest">
          <wsdlsoap:body use="literal"/>
        </wsdl:input>
        <wsdl:output name="getVendorVersionResponse">
          <wsdlsoap:body use="literal"/>
        </wsdl:output>
        <wsdl:fault name="ImplementationExceptionFault">
          <wsdlsoap:fault name="ImplementationExceptionFault" use="literal"/>
        </wsdl:fault>
      </wsdl:operation>
    </wsdl:binding>

    <!-- ALEACSERVICE -->
    <wsdl:service name="ALEACService">
      <wsdl:port binding="impl:ALEACServiceBinding" name="ALEACServicePort">
        <!-- The value of the location attribute below is an example only;
             Implementations are free to choose any appropriate URL. -->
        <wsdlsoap:address location="http://localhost:8080/services/ALEACService"/>
      </wsdl:port>
    </wsdl:service>
  </wsdl:definitions>
```

5 Bindings for the Reading and Writing Callback APIs

This section specifies XML-based bindings for the `ALECallback` and `ALECCCallback` interfaces, through which the ALE Reading API and the ALE

Writing API, respectively, deliver asynchronous notifications to subscribers. Each binding of these interfaces specifies a syntax for notification URIs. A notification URI is supplied by an ALE client as a parameter of the `subscribe` and `unsubscribe` methods of the ALE Reading or Writing API. The notification URI both selects a binding of the callback interface to be used for that subscriber, and provides addressing information in a manner specified by each binding below.

Each subsection below specifies the conformance requirement (MAY, SHOULD, SHALL) for each binding. Implementations MAY provide additional, vendor-specific bindings of the callback interfaces. If an implementation provides an additional binding of the callback interface, it SHALL use a URI scheme that does not conflict with any of the standardized bindings.

All notification URIs recognized by bindings as legal, whether the binding is standardized as a part of this specification or not, SHALL conform to the general syntax for URIs as defined in [RFC2396]. Each binding may impose additional constraints upon syntax.

A given mechanism for delivery of asynchronous results may be used in a binding of the `ALECallback` (Reading API) interface and the `ALECCCallback` (Writing API) interface. This specification defines bindings of the `ALECallback` and `ALECCCallback` for each of four delivery mechanisms: HTTP, raw TCP, File, and HTTP over TLS (HTTPS). Because the specifications of the `ALECallback` binding and the `ALECCCallback` binding for a given delivery mechanism are nearly identical, they are pairwise combined in the specifications below.

5.1 HTTP Bindings

The HTTP bindings of the `ALECallback` and `ALECCCallback` interfaces provide for delivery of `ECReports` or `CCReports`, respectively, in XML via the HTTP protocol using the POST operation. Implementations SHOULD provide support for these bindings.

The syntax for HTTP notification URIs as used by these bindings is defined in [RFC2616], Section 3.2.2. Informally, an HTTP URI has one of the two following forms:

```
http://host:port/remainder-of-URL
http://host/remainder-of-URL
```

where

- *`host`* is the DNS name or IP address of the host where the callback receiver is listening for incoming HTTP connections.
- *`port`* is the TCP port on which the callback receiver is listening for incoming HTTP connections. The port and the preceding colon character may be omitted, in which case the port defaults to 80.
- *`remainder-of-URL`* is the URL to which an HTTP POST operation will be directed.

The ALE implementation delivers event cycle or command cycle reports by sending an HTTP POST request to the callback receiver designated in the URI, where *`remainder-of-URL`* is included in the HTTP `request-line` (as defined in [RFC2616]), and where the payload is the `ECReports` instance or `CCReports` instance encoded in XML according to the schema specified in Section 3.5 or Section 3.6, respectively.

The interpretation by the ALE implementation of the response code returned by the callback receiver is outside the scope of this specification; however, all implementations SHALL interpret a response code 2xx (that is, any response code between 200 and 299, inclusive) as a normal response, not indicative of any error.

5.2 TCP Bindings

The TCP bindings of the `ALECallback` and `ALECCCallback` interfaces provide for delivery of `ECReports` or `CCReports`, respectively, in XML via a raw TCP connection. Implementations SHOULD provide support for these bindings.

The syntax for TCP notification URIs as used by these bindings is as follows:

```
tcp_URL = "tcp:" "//" host ":" port
```

where the syntax definition for `host` and `port` is specified in [RFC2396].

Informally, a TCP URI has the following form:

`tcp://`*`host`*`:`*`port`*

The ALE implementation delivers an event cycle or command cycle report by opening a new TCP connection to the specified host and port, writing to the connection the `ECReports` instance or `CCReports` instance encoded in XML according to the schema specified in Section 3.5 or Section 3.6, respectively, and then closing the connection. The ALE implementation SHALL NOT require a reply or acknowledgement.

5.3 FILE Bindings

The FILE bindings of the `ALECallback` and `ALECCCallback` interfaces provide for writing of `ECReports` or `CCReports`, respectively, in XML to a file. Implementations MAY provide support for these bindings.

The syntax for FILE notification URIs as used by these bindings is defined in [RFC1738], Section 3.10. Informally, a FILE URI has one of the two following forms:

`file://`*`host/path`*
`file:///`*`path`*

where

- *`host`* is the DNS name or IP address of a remote host whose filesystem is accessible to the ALE implementation.

- `path` is the pathname of a file within the remote filesystem, or the local filesystem if `host` is omitted.

The ALE implementation delivers an event cycle or command cycle report by appending to the specified file the `ECReports` or `CCReports` instance encoded in XML according to the schema specified in Section 3.5 or Section 3.6, respectively. Note that if more than one event cycle completes, the file will contain a concatenation of XML documents, rather than a single XML document.

Implementations of this binding may impose additional constraints on the use of the FILE URI. For example, some implementations of this binding may support only a local filesystem while others may support only a remote filesystem, some implementations of this binding may impose further restrictions on the syntax of the `path` component, and so forth. This specification also does not define the behavior when `path` names a directory; the behavior in that case is implementation dependent.

Rationale (non-normative): The intended use for the FILE bindings is for debugging, and hence the specification is intentionally lax in order to give freedom to implementations to provide the most appropriate and useful facility given the unique circumstances of that implementation.

5.4 HTTPS Bindings

The HTTPS bindings of the `ALECallback` and `ALECCCallback` interfaces provide for delivery of `ECReports` or `CCReports`, respectively, in XML via the HTTP protocol using the POST operation, secured via TLS. The HTTPS protocol provides link-level security, and optionally mutual authentication between an ALE implementation and its callback receivers. Implementations MAY provide support for these bindings.

The syntax for HTTPS notification URIs as used by these bindings is defined in [RFC2818], Section 2.4, which in turn is identical to the syntax defined in [RFC2616], Section 3.2.2, with the substitution of `https` for `http`. Informally, an HTTPS URI has one of the two following forms:

```
https://host:port/remainder-of-URL
https://host/remainder-of-URL
```

where

- `host` is the DNS name or IP address of the host where the callback receiver is listening for incoming HTTPS connections.
- `port` is the TCP port on which the receiver is listening for incoming HTTPS connections. The port and the preceding colon character may be omitted, in which case the port defaults to 443.
- `remainder-of-URL` is the URL to which an HTTP POST operation will be directed.

The ALE implementation SHALL deliver event cycle or command cycle reports by sending an HTTP POST request to the callback receiver designated in the URI, where `remainder-of-URL` is included in the HTTP `request-line` (as defined in [RFC2616]), and where the payload is the `ECReports` or `CCReports` instance encoded in XML according to the schema specified in Section 0 or Section 3.6, respectively.

For these bindings, HTTP SHALL be used over TLS as defined in [RFC2818]. TLS for this purpose SHALL be implemented as defined in [RFC2246] except that the mandatory cipher suite is `TLS_RSA_WITH_AES_128_CBC_SHA`, as defined in [RFC3268] with CompressionMethod.null. Implementations MAY support additional cipher suites and compression algorithms as desired.

The interpretation by the ALE implementation of the response code returned by the callback receiver is outside the scope of this specification; however, all implementations SHALL interpret a response code 2xx (that is, any response code between 200 and 299, inclusive) as a normal response, not indicative of any error.

Warning (non-normative): The HTTPS report delivery mechanism is concerned only with the delivery of ECReports and CCReports from the ALE implementation to subscribed receivers. Using this mechanism is more secure than using cleartext protocols such as HTTP or "raw" TCP but it does not ensure that the system as a whole is secure. In a typical RFID system there will be other communications paths (for example reader-to-ALE, and vendor-specific ALE administrative interfaces) that would also need to be secure in order to claim that the entire system was secure.

6 Appendix: Schema and WSDL Differences from ALE 1.0 (non-normative)

This section enumerates differences between the XSD/WSDL for the ALE 1.1 Reading API and the XSD/WSDL for ALE 1.0.

6.1 Fully Compatible Changes

The changes below are fully compatible, in that a document conforming to the ALE 1.1 schema is valid according to the ALE 1.0 schema, and vice versa.

- In the definition of `ECReportGroupList`, an `anyAttribute` declaration is added. This is part of the extensibility mechanism defined in Section 3.2, but was inadvertently omitted from the ALE 1.0 schema.
- In the definition of `ECReportOutputSpec`, an `anyAttribute` declaration is added. This is part of the extensibility mechanism defined in Section 3.2, but was inadvertently omitted from the ALE 1.0 schema.
- Several types have been extended by adding new subelements to the `<extension>` element, following the extensibility mechanism defined in Section 3.2. A nested `<extension>` element is also added, in order to provide for further extensions in later versions of the specification. The types that have been extended in this way include: `ECBoundarySpec`, `ECFilterSpec`, `ECGroupSpec`,

ECReportGroupListMember, ECReportOutputSpec, ECReportSpec, and ECSpec.

- The ECReport type has been extended by the addition of a new, optional attribute, following the extensibility mechanism defined in Section 3.2.
- The following new types have been added to the schema: ECInitiationCondition, ECReportMemberField, ECReportOutputFieldSpec, ECStatProfileName, ECTagStat, ECTagTimestampStat, ECFieldSpec, ECFilterListMember, ECIncludeExclude, ECReaderStat, and ECSightingStat.

6.2 Non-Forward Compatible Changes

The changes below are backward compatible, meaning that a document conforming to the ALE 1.0 schema is valid according to the ALE 1.1 schema, but not forward compatible, meaning that a document conforming to the ALE 1.1 schema may not be valid according to the ALE 1.0 schema in some cases. The lack of forward compatibility stems from areas of non-extensibility in the ALE 1.0 schema.

- The ECGroupSpec type has been changed to include the extensibility mechanism, and the extensibility mechanism is used to add a new, optional subelement. An ECGroupSpec that includes the new, optional fieldspec subelement will not be valid according to the ALE 1.0 schema.
- All enumeration types have been made extensible in ALE 1.1, by changing the schema to accept any string. Consequently, a document that contains an enumeration value not specified in ALE 1.0 will not be valid according to the ALE 1.0 schema. The enumeration types are: ECReportSetEnum, ECTerminationCondition, and ECTimeUnit, along with ImplementationExceptionSeverity defined in the WSDL.

6.3 Corrections

Two errors in the ALE 1.0 schema have been corrected in the ALE 1.1 schema.

- In the ECReportSetSpec type, the set attribute is now declared to be required. In the ALE 1.0 schema, the use declaration was omitted which implied that the attribute was optional.
- In the ECTime type, the unit attribute is now declared to be required. In the ALE 1.0 schema, the use declaration was omitted which implied that the attribute was optional.

Technically, these changes are not backward compatible, in that a document that conforms to the ALE 1.0 schema may not be valid according to the ALE 1.1 schema. However, this only occurs if the document omitted one of the two attributes listed above. The main body of the ALE 1.0 specification did not specify that these attributes are optional, and no behavior was defined for the case where they are omitted. Therefore, the lack of back-compatibility only occurs for a document whose meaning is not defined according to the ALE 1.0 specification, and so the lack of compatibility is unlikely to arise in practice.

6.4 Changes in Idiom

The following changes to the schema do not affect what documents are valid or invalid according to the schema, but simply changes the XSD description to an alternative, equivalent form.

- The `ECTrigger` type, which is equivalent to `xsd:string`, is now defined as a (trivial) restriction of `xsd:string` rather than as a (trivial) extension of `xsd:string`. This allows `ECTrigger` to be used as the type of an attribute as well as the type of an element.
- Several type names were defined in the ALE 1.0 schema in places where the UML specified a field of type `List<…>`. These type names appeared only in the schema, and not in the UML. To avoid confusion, in the ALE 1.1 schema these type names are eliminated by embedding the type definition in the context where the reference to the list type occurs. The type names that have been eliminated in this way are: `ECExcludePatterns`, `ECIncludePatterns`, `ECLogicalReaders`, `ECReportList`, and `ECReportSpecs`.

7 References

[ALE1.1Part1] EPCglobal, "The Application Level Events (ALE) Specification, Version 1.1.1 Part I: Core Specification," EPCglobal Ratified Standard, March 2009, http://www.epcglobalinc.org/standards/ale/ale_1_1_1-standard-core-20090313.pdf.

[ISODir2] ISO, "Rules for the structure and drafting of International Standards (ISO/IEC Directives, Part 2, 2001, 4th edition)," July 2002.

[RFC1738] T. Berners-Lee, L. Masinter, M. McCahill, "Uniform Resource Locators (URL)," RFC 1738, December 1994, http://www.ietf.org/rfc/rfc1738.

[RFC2246] T. Dierks, C. Allen, "The TLS Protocol, Version 1.0," RFC2246, January 1999, http://www.ietf.org/rfc/rfc2246.

[RFC2396] T. Berners-Lee, R. Fielding, L. Masinter, "Uniform Resource Identifiers (URI): Generic Syntax," RFC2396, August 1998, http://www.ietf.org/rfc/rfc2396.

[RFC2616] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1," RFC2616, June 1999, http://www.ietf.org/rfc/rfc2616.

[RFC2818] E. Escorla, "HTTP Over TLS," RFC2818, May 2000, http://www.ietf.org/rfc/rfc2818.

[RFC3268] P. Chown, "Advanced Encryption Standard (AES) Cipersuites for Transport Layer Security (TLS)," RFC3268, June 2002, http://www.ietf.org/rfc/rfc3268.

[SOAP1.1] D. Box, D. Ehnebuske, G. Kakivaya, A. Layman, N. Mendelsohn, H. F. Nielsen, S. Thatte, and D. Winer, “Simple Object Access Protocol (SOAP) 1.1,” W3C Note, May 2000, http://www.w3.org/TR/2000/NOTE-SOAP-20000508/.

[WSDL1.1] E. Christensen, F. Curbera, G. Meredith, S. Weerawarana, “Web Services Description Language (WSDL) 1.1,” W3C Note, March 2001, http://www.w3.org/TR/2001/NOTE-wsdl-20010315.

[WSI] K. Ballinger, D. Ehnebuske, M. Gudgin, M. Nottingham, P. Yendluri, “Basic Profile Version 1.0,” WS-i Final Material, April 2004, http://www.ws-i.org/Profiles/BasicProfile-1.0-2004-04-16.html.

[XML1.0] T. Bray, J. Paoli, C. M. Sperberg-McQueen, E. Maler, F. Yergeau, “Extensible Markup Language (XML) 1.0 (Third Edition),” W3C Recommendation, February 2004, http://www.w3.org/TR/2004/REC-xml-20040204/.

[XSD1] H. Thompson, D. Beech, M. Maloney, N. Mendelsohn, “XML Schema Part 1: Structures,” W3C Recommendation, May 2001, http://www.w3.org/TR/xmlschema-1/.

[XSD2] P. Biron, A. Malhotra, “XML Schema Part 2: Datatypes,” W3C Recommendation, May 2001, http://www.w3.org/TR/xmlschema-2/.

[XMLVersioning] D. Orchard, “Versioning XML Vocabularies,” December 2003, http://www.xml.com/pub/a/2003/12/03/versioning.html.

8 Credits

See [ALE1.1Part1], Section 18.